DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
E INTELIGENCIA ARTIFICIAL

Modelado de Sistemas Dinámicos con Machine Learning

Aplicaciones al Mantenimiento Basado en la Condición

V. B. Director

Memoria presentada por
Diego Cabrera Mendieta
para optar al grado de Doctor
por la Universidad de Sevilla

D. Fernando Sancho Caparrini

Sevilla, Octubre de 2017.

Dedicatoria

A mi hija Sophie Romina y mi esposa María Teresa, por ser y estar. Las amo

Agradecimientos

Este trabajo fue posible gracias al apoyo y colaboración de familia, amigos y colegas a los que deseo presentar mis agradecimientos. Especialmente a Carmen y Milton, padres incansables que supieron motivar mi crecimiento personal. También a mi hermana Johanna, por darme la responsabilidad de dar el ejemplo inherente en un hermano mayor. A Fernando, por poner alma vida y corazón en la dirección de este trabajo y por compartir su conocimiento y tiempo de manera desinteresada para que este proyecto se mantenga a flote.

La madurez lograda solo pudo ser alcanzada por experiencias vividas a lo largo de este proceso. Por eso quiero agradecer también a los miembros del GIDTEC de la Universidad Politécnica Salesiana sede Cuenca, de manera especial a Mariela y Vinicio, colegas que me acompañaron en las largas jornadas de búsqueda de conocimiento en tiempo de oficina y cuyas aportaciones dieron realce a este trabajo. A Chuan y Felipe por abrirme las puertas en sus grupos de trabajo en mis estancias doctorales. A todos los miembros del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Sevilla por hacerme sentir como en casa en mis periodos lejos de casa.

Ya lo decía Jean Monnet, "Los hombres pasan, las instituciones quedan". Por eso quiero agradecer también a la Universidad Politécnica Salesiana sede Cuenca por la beca concedida para mis estudios de doctorado. A la Universidad de Chile, Universidad de Tecnología y Negocios de Chongqing y Universidad de Sevilla por recibirme en las diferentes estancias de investigación realizadas.

Resumen

El estudio de los sistemas dinámicos es un tema de gran interés tanto en la ingeniería como en las ciencias por su gran aplicabilidad a la resolución de problemas que, con frecuencia, aparecen en un sin número de áreas. Sin embargo los métodos formales hasta ahora utilizados para su análisis, no han tenido la flexibilidad suficiente para adaptarse a sistemas de complejidad creciente, donde la interacción de sus elementos no permite una inferencia directa del comportamiento del sistema, en una, o varias de sus variables. Por otro lado, los nuevos avances en las técnicas de Machine Learning han demostrado tener una gran capacidad de adaptación en dominios tan diversos resultando en la necesidad de cambios minoritarios para su aplicación entre uno u otro. A pesar de esto, su estudio en el modelado de sistemas dinámicos, como tarea fundamental, ha sido pocas veces abordado.

Por las razones anteriores, este trabajo se enfoca en el desarrollo de 3 metodologías para modelado de sistemas dinámicos desde la perspectiva del Machine Learning a partir de información incompleta de sus variables representadas como series temporales de alta complejidad. Las propuestas son presentadas en función de los diferentes escenarios de información disponible para el modelado, que pudieran llegar a aparecer en situaciones reales. Como primera metodología se propone el modelamiento del sistema dinámico con un enfoque manual de caracterización de estados usando el conocimiento a-priori del sistema mediante la descomposición por Wavelet Packet y la posterior identificación de patrones mediante una técnica clásica de clasificación llamada Random Forest. Como segunda propuesta se presenta el aprendizaje no-supervisado del proceso de caracterización que se adapta de forma automática al sistema dinámico en estudio mediante el modelo Stacked Convolucional Autoencoder, el cual inicializa una Red Neuronal Convolucional Profunda que luego es optimizada de forma supervisada, donde además el proceso de identificación de patrones se encuentra embebido, y es optimizado junto con el modelo de extracción de características. La tercera propuesta en cambio cumple la tarea de caracterización e identificación de patrones de forma no-supervisada, lo primero mediante el aprendizaje de una representación óptima de las series temporales codificada en los parámetros de un Echo State Network, y lo segundo por medio

de un Variational Autoencoder, un modelo capaz de aproximar la distribución de probabilidad (a menudo compleja) de los datos. En esta última aproximación se elimina la necesidad de conocer la etiqueta de las series de tiempo que provienen de los estados del sistema dinámico.

Las metodologías propuestas son evaluadas en tareas de mantenimiento basado en la condición como son el diagnóstico de fallos, la estimación de la severidad de daño y la detección de fallos en elementos de maquinaria rotativa (concretamente distintos tipos de engranajes y rodamientos). Los altos indices de exactitud obtenidos en los resultados de la evaluación en cada tarea, muestran que las metodologías aportadas dan un nivel elevado de confiabilidad, robustez y flexibilidad. Además, frente a comparaciones realizadas con otras metodologías reportadas en el estado del arte, las propuestas presentan un desempeño superior en todos los casos.

Índice general

Índice de figuras

CAPÍTULO 1

INTRODUCCIÓN

La curiosidad inherente del ser humano ha provocado un deseo de comprensión de los procesos que le rodean. Entre ellos se encuentran algunos que nacen de fenómenos naturales, como el interés por entender el comportamiento climático, y otros de fenómenos sociales, como los procesos de migración entre países. Junto a ellos también están los procesos que nacen del dominio que ha ejercido el ser humano sobre determinadas áreas, por ejemplo durante el desarrollo industrial, donde busca tener un conocimiento extenso de los componentes que intervienen en los procesos artificiales generados y sus posibles interacciones. En estos casos, el objetivo sobrepasa el mero conocimiento curioso y crece con la necesidad de controlar las diversas etapas de los procesos para optimizar resultados. De forma genérica, e independientemente del proceso al que dan lugar, las diversas mediciones que se pueden realizar del sistema, y algunas veces controlar, se conocen como *variables*.

Como parte de la exploración y comprensión de un proceso se buscan formas de cuantificar los eventos asociados a sus variables. Dependiendo de su complejidad, estos eventos podrían estar asociados a una sola variable o a múltiples de ellas, y a menudo siguen una evolución ordenada secuencial que podría converger en uno o varios patrones que reflejan, y a veces determinan, su comportamiento. Entendemos por *evolución ordenada secuencial* a sucesos que ocurren uno a continuación de otro, tomando como referencia una o varias dimensiones espaciales y/o la dimensión temporal. En este último caso, en el que la dimensión que marca el orden de ocurrencia es el tiempo, estaremos hablando de *eventos temporales*.

De acuerdo con ello, si un proceso genera eventos ordenados secuencialmente la forma natural de capturarlos es mediante mediciones que también siguen el mismo orden, lo que da como resultado un conjunto ordenado que, en el caso de mediciones a lo largo del tiempo, es llamado *serie de tiempo*.

En consecuencia, una serie de tiempo es una sucesión finita o infinita de elementos que guardan un orden cronológico específico entre sí. Estos elementos pueden ser el producto de un proceso de medición como el mencionado anteriormente o pueden ser generados de forma totalmente sintética. Además, dependiendo de las

características de la variable medida, estos elementos podrían ser representados en un espacio de una dimensión o de varias dimensiones. Por ejemplo, si se desea analizar el comportamiento motor de una persona comúnmente se obtendrá una serie de tiempo de la medición de la variable aceleración, que por defecto tiene tres componentes (una en cada uno de los ejes de movimiento), lo que da como resultado una serie de tiempo multi-dimensional. Si, por el contrario, se desea analizar el movimiento rectilíneo de una partícula, lo más habitual será obtener una serie de tiempo uni-dimensional. Desde el punto de vista del dato almacenado la diferencia en la representación de cada elemento radica en que el caso uni-dimensional vendrá dada por un escalar, mientras que el caso multi-dimensional vendrá dada por una tupla ordenada de elementos con estructura vectorial.

La complejidad de una serie de tiempo está directamente relacionada con el proceso que la produce[93], y se asocia usualmente con la variabilidad del proceso y la dificultad que se tiene para encontrar un patrón reconocible en su comportamiento que pueda servir para realizar inferencias sobre el proceso que la origina.

En este trabajo nos enfocamos en el análisis y modelado de series de tiempo con alta variabilidad. La perspectiva desde la cual se aborda el problema nace de la búsqueda de patrones mediante el análisis y procesamiento de señales usando el conocimiento experto a priori del proceso, hasta evolucionar a una metodología de modelado totalmente no supervisada y únicamente basada en los datos disponibles de las mediciones. Con este fin se proponen tres métodos para abordar el problema del modelado de una serie de tiempo. La primera propuesta[14] utiliza técnicas conocidas de procesamiento de señales[12], que se seleccionan en base al conocimiento reportado en la literatura para la extracción de características representativas (con patrones más notorios) de la serie de tiempo, y que luego serán usadas para la construcción de un modelo de clasificación usando el modelo Random Forest[10] de Aprendizaje Automático. La segunda propuesta[13] da un paso adelante en relación a la búsqueda de independencia del conocimiento del proceso, dando como resultado un método que evalúa la posibilidad de usar extracción no supervisada de características con el uso de un modelo de aprendizaje basado en datos llamado Stacked Convolutional Autoencoder[68], que desde una representación de la serie de tiempo en 2 dimensiones, obtenida con técnicas de procesamiento de señales, es capaz de extraer características útiles que posteriormente se usarán para construir un modelo de predicción. Estos dos enfoques iniciales presentan el problema de necesitar series de tiempo que se encuentren etiquetadas para la construcción del modelo de predicción. En consecuencia, aunque el proceso de extracción de características sea no supervisado, como en el segundo caso, bajo este enfoque no es posible conseguir independencia del conocimiento a priori en los datos. Como solución a este problema se presenta la tercera propuesta, donde se elimina totalmente la dependencia de las técnicas de procesamiento de señales y se plantea un enfoque basado en Echo State Networks[41] para proyectar de forma no supervisada la serie de tiempo a un nuevo espacio de representación, desde el cual se realiza un aprendizaje de la distribución de probabilidad asociada al proceso[25], eliminando, bajo ciertas suposiciones, la dependencia del conocimiento de la etiqueta en los datos.

Los procesos en los que nos enfocaremos para aplicar las propuestas anteriores pertenecen a un área de la ingeniería mecánica y de procesos llamada *Mantenimiento basado en la Condición* (CBM)[42], que se centra en el estudio de técnicas que ayuden a determinar el estado general de una máquina y/o sus componentes para lograr (con una buena planificación del mantenimiento) alargar su vida útil y disminuir los costos de funcionamiento. Dentro del CBM se abordan los procesos asociados al diagnóstico de fallos en engranajes y rodamientos de maquinaria rotativa, que han sido los seleccionados como objeto de estudio de este trabajo. El objetivo que buscamos para la tarea del diagnóstico de fallos es elaborar metodologías que permitan identificar daños, y/o su severidad, en cajas de engranajes y rodamientos a partir únicamente de síntomas que se puedan captar desde mediciones en variables del sistema mecánico completo. Sin duda, es deseable disponer de modelos de clasificación de fallos, y/o nivel de severidad, construidos a partir de las series de tiempo disponibles, que hagan uso de un conocimiento mínimo del proceso, y que permitan determinar el estado del componente.

Se han elegido estos componentes mecánicos porque son los más comunes, importantes y propensos a fallos en las máquinas rotativas, por lo que disponemos a la vez de suficientes datos para su análisis y del conocimiento previo necesario para verificar la robustez de las soluciones obtenidas. La variable del proceso a partir de la cual se obtienen las series de tiempo que usaremos en nuestro trabajo es la vibración de la máquina, que se obtiene por un proceso de discretización de las señales captadas por sensores llamados acelerómetros. Aunque las series de tiempo con las que trabajaremos son una representación discreta de las señales reales, los dos términos tendrán el mismo significado a lo largo de este trabajo, a menos que explícitamente se diga lo contrario.

1.1. Objetivos

Los sistemas dinámicos han sido principalmente estudiados desde el área de la teoría de control usando métodos de modelamiento clásico, sin embargo, la presencia de incertidumbre con respecto a las variables del sistema, o un desconocimiento casi total de ellas, requiere de mecanismos basados en mínimas mediciones de variables del proceso.

Teniendo en cuentas estas consideraciones, el objetivo general del trabajo que aquí se presenta es desarrollar metodologías para el modelado de sistemas dinámicos a partir de información incompleta de sus variables, representadas como series temporales de alta complejidad, y adicionalmente, mostrar su aplicación en tareas del mantenimiento basado en la condición.

Para cumplir con este objetivo general hemos visto necesario cumplir con los siguientes objetivos específicos:

- Estudiar la viabilidad de la representación de un sistema dinámico con carac-

terísticas extraídas de una serie temporal muestreada desde una sola variable de estado mediante una descomposición clásica por la Transformada de Wavelet Packet.

- Desarrollar una metodología basada en el Análisis de Señales y Aprendizaje Automático clásico para la creación de modelos predictivos de Sistemas Dinámicos.
- Presentar una metodología para la generación de modelos de extracción de características adaptables de forma automática que sean capaces de representar la dinámica del sistema mediante técnicas de Deep Learning.
- Comparar de forma experimental la propuesta anterior con un conjunto de metodologías de extracción de características reportadas en la literatura.
- Desarrollar un método de modelado de Sistemas Dinámicos basado en la reconstrucción de la señal extraída de una variable del sistema.
- Analizar la viabilidad de representar la dinámica de un sistema mediante otro modelo dinámico parametrizado.
- Presentar una metodología para la creación de modelos generativos de detección de anomalías en Sistemas Dinámicos mediante Deep Learning e Inferencia Variacional.
- Evaluar el desempeño de todas las propuestas en aplicaciones de diagnóstico de fallos, análisis de severidad de daño y/o detección de fallos en maquinaria rotativa.

1.2. Estructura de la memoria

Pasemos a ver con uno poco más de detalle las diversas propuestas metodológicas que se presentan en esta memoria para abordar este problema de modelado y análisis de sistemas dinámicos y su aplicación a tareas del mantenimiento basado en la condición.

1.2.1. Capítulo 2: Fundamentos

En el capítulo de Fundamentos se tocan cinco puntos clave que sirven como base para el desarrollo de este trabajo: *sistemas dinámicos*, *elementos mecánicos* y *maquinaria rotativa*, *sistemas de adquisición de señales*, *análisis tiempo-frecuencia* y *machine learning*. En la sección de sistemas dinámicos se dan algunas definiciones formales relativas a estos sistemas, y además se presentan informalmente algunas ideas y resultados como el teorema de Takens[93], fundamental para la justificación de este trabajo.

A pesar de que el trabajo que aquí presentamos tiene una componente teórica importante, su carácter aplicado a tareas del mantenimiento basado en la condición hace necesario abordar las características de los sistemas mecánicos (elementos, funcionamiento, posibles daños y configuraciones experimentales) con los que se van a trabajar. Es por ello que se han incluido las secciones de componentes mecánicos, fallos en elementos mecánicos y configuración experimental para tareas del mantenimiento basado en la condición. Además, en la sección de sistemas de adquisición de datos se presenta una forma adecuada desde el punto de vista tecnológico para la obtención de datos en forma de series temporales de mediciones sobre variables concretas de los sistemas mecánicos, y que será la implementada aquí.

El entorno en el que se desarrolla la teoría y la práctica de este trabajo es la frontera entre el procesamiento de señales y el aprendizaje automático. En este sentido, inicialmente se usan técnicas de procesamiento de señales clásicas para poder aplicar algoritmos de aprendizaje automático, pero posteriormente se hará uso de técnicas más elaboradas de aprendizaje como herramienta para procesar las señales de forma directa, razón por la que se ha considerado conveniente introducir en este capítulo las secciones de análisis tiempo-frecuencia de señales, y una breve introducción al aprendizaje automático.

1.2.2. Capítulo 3: Extracción de características y clasificación con un enfoque clásico

Para la creación de un modelo de clasificación basado en datos es necesario que los datos de entrada presenten de forma clara patrones reconocibles que permitan discriminar la pertenencia a una clase. En el contexto de la clasificación a estos datos de entrada se les conoce como características. Por ejemplo, en aplicaciones de diagnóstico de fallos en maquinaria rotativa se desea conocer el fallo específico que tiene un componente a partir de un conjunto de entradas. Sin embargo, para los procesos que son fuente de estudio de este trabajo, los elementos de una serie de tiempo no pueden ser vistos directamente como características por su gran complejidad y porque los patrones identificables no permanecen fijos a lo largo del tiempo a pesar de tratarse de mediciones del mismo proceso bajo las mismas condiciones.

Por ello, en esta primera propuesta se detalla un método que empieza con una etapa de extracción de características a partir de series de tiempo utilizando la técnica de descomposición por Wavelet Packet (WPD)[31]. Para esto, y como se detallará más adelante, se calcula de cada señal su WPD hasta el sexto nivel de descomposición, obteniendo finalmente de cada señal una colección de sub-señales llamadas *coeficientes de descomposición*. A su vez, para cada coeficiente obtenido del proceso anterior se calcula su energía. Todas estas energías calculadas de la descomposición de la WPD de la señal son agrupadas formando un vector de características representativas de dicha señal. Por otro lado, como la WPD depende de una wavelet madre específica, y cada una de ellas puede aportar información distinta a las demás, se han elegido cinco de ellas para su evaluación: Daubechies 7 (db7), Symlet 3 (sym3),

Coiflet 4 (coif4), Biorthogonal 6.8 (bior) y Reverse Biorthogonal (rbior). Así, se obtiene una representación compacta de 320 características (64 características de cada una de las 5 wavelets madre) de una serie de tiempo que inicialmente puede tener miles de elementos, dependiendo del tiempo de adquisición y la tasa de muestreo.

Con esta nueva representación de las señales se construye un modelo Random Forest[10] para la etapa de clasificación que es capaz de discriminar señales a partir de sus características representativas. Como veremos en el capítulo, este modelo, por ejemplo, es capaz de decidir si una señal de vibración representada por un conjunto de características fue extraída de una máquina que tiene un fallo en la pista interna de un rodamiento, o, por el contrario, que esa señal representa a una máquina sin fallo en ningún componente.

El método propuesto es evaluado para tres aplicaciones de CBM. La primera es el diagnóstico de fallos en cajas de engranajes rectos en donde se tienen 7 clases representando 7 diferentes estados de un engranaje. La segunda es la determinación de la severidad de un daño por ruptura de diente en una caja de engranajes helicoidales, donde se tienen 10 clases que representan los niveles de severidad de ruptura de diente en el piñón de una caja reductora. La tercera aplicación también es de severidad de daño, pero en este caso para los componentes de un rodamiento (pista externa, pista interna y elemento rodante) dispuesto en un sistema rotativo de transmisión de movimiento. Para cada una de las aplicaciones se realiza una búsqueda exhaustiva sobre los diferentes hiperparámetros propios del modelo de Random Forest con el fin de encontrar los valores más adecuados para ellas, mostrando que el método propuesto puede ser aplicado de forma satisfactoria.

Sin embargo, esta primera aproximación tiene dos debilidades fundamentales: por una parte, requiere de un proceso manual de extracción de características basado en conocimiento experto, y por otra, necesita la información de las clases en las que se pueden clasificar las señales para generar el modelo de clasificación.

1.2.3. Capítulo 4: Aprendizaje de características de series de tiempo

Para mitigar las debilidades de la propuesta anterior presentamos una fusión entre la extracción de características y el modelo de predicción. De esta forma, el método propuesto ya no necesita técnicas de procesamiento de señales basadas en el conocimiento del proceso y se logra su automatización.

Con este fin se hará uso de Redes Convolucionales Profundas (DCNN)[32] como extractor de características. Este modelo tiene como base fundamental el aprendizaje de un conjunto de *kernels de convolución* que se aplican a un grupo de funciones discretas en 2 dimensiones. El resultado de esta aplicación es otro grupo de funciones, llamados *mapas de características*, a las que nuevamente se aplica el proceso de convolución en una capa posterior, obteniendo un modelo que crece en profundidad de acuerdo al número deseado de capas de convolución. En cada nivel de iteración, los mapas de características obtenidos proporcionan una representación

de la información de entrada que es más abstracta que la de la capa anterior, resultando en la extracción de características representativas. El proceso de aprendizaje de los kernels de convolución está basado en una modificación del algoritmo de retropropagación del error para el cálculo del gradiente habitual en las redes neuronales artificiales clásicas. Sin embargo, se sabe que este algoritmo de optimización tiene el problema de que el gradiente tiende rápidamente a anularse cuando la red es profunda (tiene muchas capas), por lo que el entrenamiento se hace inviable ya que no se produce un aprendizaje significativo. En este trabajo se propone solventar este problema mediante un pre-entrenamiento de cada capa convolucional con un modelo denominado Convolutional Autoencoder[68] (CAE, una modificación para una capa convolucional del conocido Autoencoder Clásico).

Sin embargo, una DCNN en su forma tradicional necesita que la entrada a la red sea una función binaria (habitualmente, una imagen), por lo que realizamos un preprocesamiento que consiste en transformar cada serie de tiempo, que se encuentra en una sola dimensión, en una representación bidimensional. Para ello, haremos uso nuevamente del cálculo de la WPD de cada serie de tiempo hasta el sexto nivel de descomposición de una wavelet madre daubechies 5 (db5), obteniendo los coeficientes de descomposición, que al ser el resultado de un proceso recursivo de aplicación de filtros espejo en cuadratura representan rangos específicos de componentes frecuenciales que pueden ser ordenados de menor a mayor para construir una representación tiempo-frecuencia bidimensional de la serie de tiempo.

Las características que se extraen con la DCNN sirven de entradas para un modelo de clasificación. En nuestro caso el modelo elegido es una red *feedforward*. Su entrenamiento se realiza en conjunto con la DCNN por lo que no es posible separar los dos modelos, lo que provoca que las características se especialicen de acuerdo a la tarea específica de clasificación, en vez de ser éstas de propósito general como en la primera propuesta. Sin embargo, sigue teniendo el inconveniente de que la extracción de características es dependiente de la información de la clase a la que pertenecen las instancias de entrenamiento. Así pues, este método elimina una de las debilidades de la propuesta anterior, pero sigue presentando la otra.

El método propuesto es evaluado para el diagnóstico de la severidad de un fallo por ruptura de diente en engranajes helicoidales bajo condiciones de funcionamiento estacionarias y no-estacionarias, que también fue analizado con el método presentado en el capítulo anterior. Además, se realiza una comparación entre la DCNN con y sin el proceso de pre-entrenamiento para diferentes niveles de profundidad de la red, mostrando que el proceso de pre-entrenamiento con SCAE tiene un efecto positivo en el desempeño de la red para esta tarea. También se muestra una comparación exhaustiva con otros métodos supervisados y no supervisados de extracción de características reportados en la literatura, constatando que los resultados obtenidos en el método aquí propuesto presenta mejor desempeño que las demás opciones encontradas.

1.2.4. Capítulo 5: Aprendizaje de la representación y one-class learning

Finalmente, en nuestra última propuesta presentamos un método no supervisado, tanto para la extracción de características como para el aprendizaje del modelo capaz de discriminar instancias. Para la extracción de características se utiliza un modelo de *Computación por Reservorios* (RC)[41], y para la discriminación de clases se reemplazan los modelos de aprendizaje supervisado por el modelado no supervisado de la distribución de probabilidad del proceso mediante un *Autoencoder Variacional* (VAE)[25].

Los modelos de RC están basados en *Redes Recurrentes* (RNN) que tienen como parte fundamental de su arquitectura una sección, llamada *reservorio*, encargada de proyectar una entrada con dependencia temporal entre sus elementos a un conjunto de estados. El modelo concreto de RC elegido para este trabajo es el de *Echo State Network* (ESN) que restringe los estados con ciertas propiedades que garantizan la estabilidad de la red. En general, los modelos de RC no ajustan los parámetros del reservorio, solamente se aprenden las conexiones que van desde el reservorio hasta una capa de salida, que se denominan *pesos de la periferia*, y que, como problema de optimización, puede ser resuelto mediante el algoritmo de *ridge regression*[67]. En este capítulo se propone el uso de una ESN para descomponer la serie de tiempo de entrada en un conjunto de estados, desde los cuales se optimizan los pesos de la periferia para predecir el siguiente instante de la serie de tiempo. Podemos interpretar que estos pesos capturan una representación estática de la señal, que es usada como su vector de características, por lo que podemos decir que hemos usado este modelo para conseguir una codificación estática de la señal dinámica temporal.

Con las características obtenidas se modelan las series de tiempo que pertenecen a un mismo proceso (por ejemplo, señales obtenidas en una condición específica de la máquina). Este proceso se realiza con un VAE que aprende una aproximación a la función no-lineal y que permite transformar muestras de una distribución simple como la gaussiana en elementos de la distribución asociada al proceso que deseamos modelar. El modelo final es capaz de generar instancias de la representación de las series de tiempo desde un conjunto de instancias obtenidas por un proceso de muestreo de la distribución simple.

Basándonos en este resultado, se propone una métrica para la discriminación de nuevas instancias de series de tiempo. Esta métrica usa el proceso de codificación de VAE, que normalmente es desechado en la mayoría de los usos reportados en la literatura, y el error de reconstrucción de la entrada. Para el primero se obtiene la proyección en un espacio simplificado de la posible distribución de probabilidad desde la cual se pudo haber extraído la instancia que se prueba, para luego compararla con la distribución gaussiana aprendida. Para el segundo se reconstruye la instancia desde la distribución gaussiana y se compara con la instancia original.

Para validar el modelo, el método ha sido evaluado en la detección de fallos tanto de cajas de engranajes helicoidales como de rodamientos. En estas aplicaciones, a diferencia de las presentadas anteriormente, se toma en cuenta la dificultad de

obtener instancias de entrenamiento de una máquina con daños (ya que hemos de disponer de una máquina dañada para ello, y nuestro objetivo es salvaguardar el funcionamiento de las mismas), y sin embargo, en la mayoría de casos resulta sencillo obtener instancias en condiciones normales. Partiendo de estas consideraciones, las aplicaciones para la detección de fallos buscan tener la capacidad de reconocer que una máquina contiene algún componente que trabaja en condiciones alejadas de lo normal. Nuestra propuesta permite crear un modelo de detección de fallos que solamente necesita de instancias en condiciones normales en su fase de entrenamiento. Las pruebas se realizaron en condiciones de operación de velocidad y carga constante y variable en los mecanismos del sistema, y los resultados muestran que el modelo es capaz de discriminar al $100\,\%$ las instancias en condiciones normales de las instancias que muestran algún tipo de fallo en las dos aplicaciones.

1.2.5. Capítulo 6: Conclusiones

Finalmente, y en base a los resultados obtenidos en los capítulos anteriores, en este capítulo se presentan las conclusiones generales del trabajo realizado, tanto desde una perspectiva específica a cada una de las propuestas estudiadas, como desde una perspectiva global que aúna y compara todas estas propuestas. También se muestran algunas posibles líneas de investigación futuras en torno a la temática aquí desarrollada y a los buenos resultados obtenidos, evaluando el modelado de sistemas dinámicos con Machine Learning, esperando motivar esta vía como una de las posibles líneas de trabajo complementaria al resto de aproximaciones existentes en la literatura.

CAPÍTULO 2

FUNDAMENTOS

En este capítulo se presentan los fundamentos correspondientes a los temas transversales de la memoria, como son: Sistemas Dinámicos, elementos mecánicos y maquinaria rotativa, sistemas de adquisición de señales, análisis tiempo-frecuencia y Aprendizaje Automático.

En la sección de Sistemas Dinámicos se proporcionarán algunas definiciones básicas del área, así como una formalización de qué entenderemos por *Sistema Dinámico*. Además mostraremos el *Teorema de Takens*[93], elemento fundamental para la justificación de este trabajo.

El carácter aplicado de este trabajo de tesis en tareas del mantenimiento basado en la condición hace necesario abordar ciertos fundamentos importantes de los sistemas mecánicos y las posibles fallas que en estos pueden ocurrir. Es por ello que se han incluido las secciones de componentes mecánicos, fallos en elementos mecánicos y configuración experimental para tareas del mantenimiento basado en la condición. Adicionalmente, y con el objetivo de contextualizar adecuadamente la metodología completa de trabajo que se ha seguido, se presenta el sistema de adquisición de señales propuesto.

Como hemos indicado, este trabajo se desarrolla en la frontera entre el Procesamiento de Señales y el Aprendizaje Automático. En este sentido, inicialmente se usan técnicas de procesamiento de señales clásicas para poder aplicar algoritmos de aprendizaje automático, pero luego se utiliza el aprendizaje automático como herramienta para procesar las señales de forma directa. Por las razones anteriores se ha visto conveniente incluir en este capítulo las secciones de Análisis tiempo-frecuencia y Aprendizaje Automático para presentar los fundamentos de estas dos importantes áreas.

2.1. Sistemas Dinámicos

Los fenómenos observables, ya sean naturales o derivados de procesos generados por el ser humano, dan como resultado un comportamiento que está íntimamente relacionado con las características de los elementos que intervienen (incluyendo el entorno en el que tiene lugar) y de las interacciones que se producen entre ellos. Al conjunto de elementos que interaccionan y que forman una unidad para la comprensión de un fenómeno es a lo que se llama, habitualmente y dependiendo de la disciplina de aproximación, un *sistema*.

Podemos encontrar sistemas que tienen asociados comportamientos que pueden llegar a ser muy complejos, ya sea desde un punto de vista descriptivo o dinámico, es decir, cuya estructura no es fácilmente formalizable, o cuya dinámica temporal no se puede describir de forma sencilla haciendo uso de las herramientas disponibles[1]. Sin embargo, si bajamos a un nivel descriptivo suficiente de los elementos que lo constituyen, frecuentemente nos encontramos con que tanto la descripción de éstos como de las interacciones que se producen entre ellos, es relativamente simple y sí es abordable con las herramientas científicas actuales.

Lo anterior plantea una de las incógnitas fundamentales en el estudio de sistemas, y es la de explicar cómo es posible obtener un comportamiento macroscópico (a nivel del sistema completo) que puede presentar características de complejidad cualitativamente distintas a las que se presentan a nivel microscópico (a nivel de los elementos que constituyen el sistema) o incluso mesoscópico (a un nivel intermedio, de subsistemas si los hubiera, que componen el sistema global). Sin duda, la única explicación plausible es que la conjunción de estructuras y funcionalidades simples bajo la existencia de un suficiente número de interacciones (aunque también sean simples) hace que los sistemas actuales de descripción de dinámicas y estructuras no sean lo suficientemente potentes para poder explicar los comportamientos macroscópicos observados. El número y diversidad de interacciones que se pueden obtener a partir de un conjunto prefijado de elementos es exponencial en el tamaño del conjunto, por lo que precisamos de herramientas y lenguajes nuevos para poder abordar la descripción del funcionamiento del sistema completo con algunas garantías de éxito.

Cuando en un sistema se produce un cambio en el tiempo, hablamos de *Sistema Dinámico* y, en estos casos, el análisis que se realiza al estudiar la evolución del sistema según esta dimensión temporal se denomina *análisis temporal*. Dependiendo de la referencia temporal necesaria para describir la dinámica del sistema se pueden considerar sistemas dinámicos *continuos*, aquellos que hacen uso de un espacio continuo como dominio de la variable temporal (normalmente, los números reales), o *discretos*, aquellos que hacen uso de un espacio discreto para tal fin (normalmente, los números naturales o enteros).

[1] Es interesante observar que, en muchas definiciones habituales en la literatura, la complejidad de un sistema se mide en función de nuestra capacidad actual de abordarlo.

Para formalizar de una manera más adecuada el proceso de modelado por sistemas dinámicos empezaremos hablando sobre la forma de describir cada etapa de su evolución. Para ello, es necesario que el sistema sea descrito de forma completa por un conjunto de variables que llamaremos *estados*, los cuales se agrupan para formar un *vector de estados*. En la descripción de la dinámica es habitual imponer que el vector de estados contenga el grupo de variables necesario para describir el estado siguiente a uno dado desde el punto de vista temporal. Pero, además, se suelen incluir variables complementarias (no necesarias para obtener un estado siguiente) que aportan información adicional relacionada con la tarea que se está analizando.

Para facilitar la descripción formal del sistema, se suele imponer la restricción de que el vector de estados debe tener estructura fija, tanto en tamaño como en las variables que contiene, durante todo el proceso de evolución del sistema dinámico. Lo que quiere decir que no se puede agregar, quitar o cambiar variable alguna, sino que su selección e inclusión se realiza de forma única en el proceso de modelado por un experto que define las variables más importantes. De esta forma, un estado instantáneo puede ser visto como un punto de un espacio prefijado, llamado *espacio de estados*, definido por el número y tipo de elementos del vector que lo conforma. Al estado del que parte la dinámica del sistema se le conoce como *estado inicial*, y suele asociarse al valor 0 de la variable temporal (tanto en el caso discreto como en el continuo).

Por otra parte, la forma en la que el sistema evoluciona de un estado a otro suele formalizarse por medio de una regla de evolución que nosotros interpretaremos como una *función*. En este sentido, el papel que cumple esta función es la de transformar un estado en un instante de tiempo determinado (definido por el vector de estado) en otro que sea también parte del espacio de estados. Lo más habitual es considerar esta función de dos posibles tipos, *determinista* (si la regla devuelve siempre el mismo estado a partir de las mismas condiciones de entrada), o *estocástica* (si el estado que devuelve la regla viene dado por una distribución de probabilidad, por lo que puede devolver estado distintos bajo las mismas condiciones).

En el caso de un sistema dinámico determinista discreto podemos escribir su dinámica de la siguiente forma. Si $\mathbf{x}(t)$ representa el vector de estado del sistema para un tiempo t determinado, y $\mathbf{x_0}$ representa el estado inicial, entonces la dinámica viene dada por las siguientes ecuaciones:

$$\mathbf{x}(0) = \mathbf{x_0} \tag{2.1}$$

$$\mathbf{x}(t) = f(\mathbf{x}(t-1)) \tag{2.2}$$

Debemos tener en cuenta que el instante inicial, $t = 0$, simplemente indica el instante desde el cual se inicia el análisis del sistema dinámico como un valor de referencia, por lo que en la evolución de un sistema dinámico también se podría considerar $t < 0$.

En los sistemas continuos deterministas la función f no puede representar la regla de cambio de estados sucesivos, debido a que los estados cambian en un espacio continuo con respecto al tiempo t. En este caso se suelen considerar restricciones

adicionales, como son que el vector de estados sea una función continua y derivable en un espacio con suficientes propiedades. De esta forma, la función f define la tasa de cambio instantánea del vector de estados, y la dinámica se puede expresar como:

$$\mathbf{x}(0) = \mathbf{x_0} \tag{2.3}$$
$$\mathbf{x}'(t) = f(\mathbf{x}(t)) \tag{2.4}$$

El vector de estados junto con su evolución temporal puede ser usado para describir la dinámica completa del sistema dinámico. Una forma común de representar dicha dinámica es mediante trayectorias representadas gráficamente por los valores obtenidos en cada instante de tiempo de las variables de estado en el espacio cartesiano formado por dichas variables. Al diagrama resultante del proceso anterior se le conoce como *diagrama de fase*. Bajo ciertas condiciones, el análisis de la topología de las trayectorias en el diagrama de fase permite conocer cuál es el estado del sistema e incluso su posible comportamiento futuro.

De cualquier forma, en muchos casos, independientemente de si el sistema es continuo o discreto, la información correspondiente a la evolución temporal de las variables de estado es muy limitada, y en ciertas ocasiones es totalmente inaccesible, por lo que no es posible realizar el análisis en el diagrama de fase generado por las trayectorias de las variables de estado y, por lo tanto, el estado del sistema resulta desconocido. Sin embargo, Floris Takens en [93] propone un teorema que asegura que, bajo ciertas hipótesis sobre el sistema, resulta posible reconstruir la dinámica del sistema original completo a partir de mediciones de algunas de las variables involucradas y, de esta forma, conocer la evolución del mismo (en realidad, no se obtiene la evolución exacta del sistema, pero se puede reconstruir una versión simplificada del espacio de fase que, en general, tiene las mismas características topológicas que el diagrama de fase creado por las variables de estado que se conocen).

El trabajo que aquí se presenta no se enfoca en el teorema de Takens ni en los métodos subyacentes (por ejemplo, véase [91] o [106]), sin embargo, su hallazgo justifica todo intento relacionado con la estimación del estado de un sistema dinámico a partir de mediciones limitadas de series de tiempo de algunas de sus variables. En este sentido, los métodos propuestos son muy diversos, partiendo desde técnicas de procesamiento de señales hasta análisis puramente matemáticos. Aquí se exploran metodologías híbridas entre el procesamiento de señales y el aprendizaje automático que logran, a partir de una serie temporal, estimar el estado del sistema.

2.2. Componentes mecánicos

2.2.1. Engranes

Los *engranes* (o engranajes) son elementos mecánicos que se pueden conectar y son capaces de transmitir movimiento de rotación y torque, que puede disminuir o

aumentar dependiendo de la relación de transmisión, r_t, entre ellos. Por esta razón, los engranes son elementos indispensables en maquinarias rotativas que requieran de una velocidad específica.

Si tenemos dos engranes tal y como muestra la Figura 2.1, donde el primero es el engrane de entrada y el segundo es el engrane de salida, la *relación de transmisión* entre ellos se calcula como el ratio del número de dientes del primero de ellos, Z_1, y el número de dientes del segundo, Z_2:

$$r_t = \frac{Z_1}{Z_2} \tag{2.5}$$

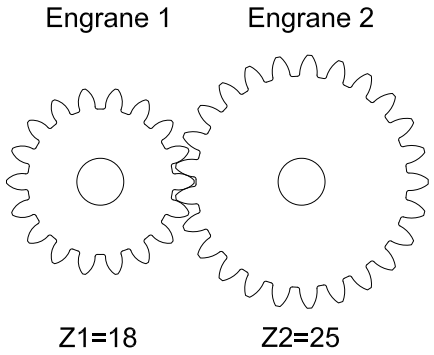


Figura 2.1: Relación de transmisión en engranes.

Cuando la relación de transmisión es menor que 1, se obtiene una disminución de la velocidad y un aumento del torque, como es el caso de las cajas reductoras. Por el contrario, si la relación de transmisión es mayor que 1, la velocidad aumentará y el torque disminuirá. Para calcular la velocidad resultante en el engrane 2 basta multiplicar la relación de transmisión entre Z1 y Z2 por la velocidad del engrane 1.

$$V_2 = \frac{Z_1}{Z_2} \cdot V_1 \tag{2.6}$$

Actualmente, en cajas reductoras se pueden utilizar engranes con diente recto o helicoidal, siendo estos últimos los más usados por su bajo ruido y la capacidad de soportar mayores velocidades de funcionamiento. La Figura 2.2 muestra una caja reductora con engranes helicoidales que cuenta con una relación de transmisión de $0{,}667$ dada por los engranes con número de dientes $Z_1 = 30$ y $Z_2 = 45$.

2.2.2. Rodamientos

Se conoce como *rodamiento* al elemento mecánico que se acopla en un eje con el objetivo de reducir la fricción existente en el eje al transmitir cargas axiales, radiales

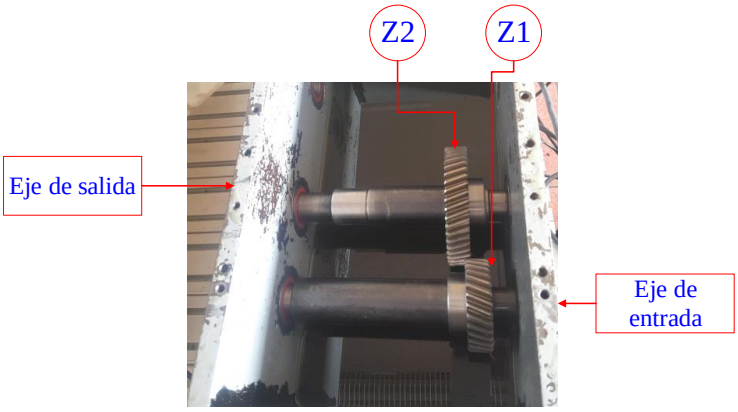


Figura 2.2: Caja reductora con engranes helicoidales.

o una combinación de ellas. Los principales componentes de un rodamiento son: pista interna, pista externa, y elemento rodante (ver Figura 2.3).

La pista interna cumple la función de acople entre eje y rodamiento, por lo tanto cuenta con un movimiento giratorio. La pista externa cumple la función de mantener al rodamiento en una posición establecida, por tanto no tiene movimiento giratorio. El elemento rodante es el encargado de soportar las cargas que actúan sobre el rodamiento y además posee un movimiento de traslación y rotación.

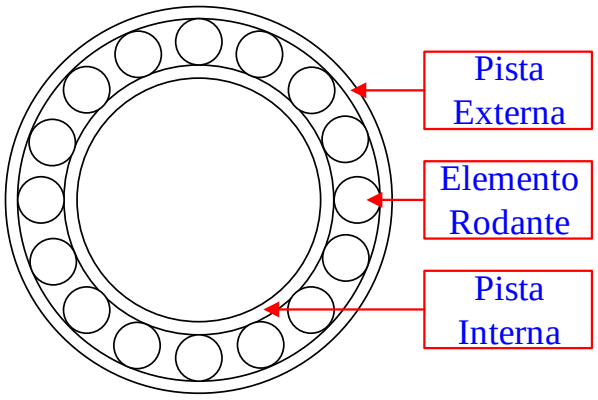


Figura 2.3: Partes principales de un rodamiento.

2.3. Fallos en elementos mecánicos

Cada pieza que forma parte de un mecanismo cumple con una tarea específica, y dependiendo de cuál sea, los elementos pueden ser sometidos a cargas axiales

o radiales, lo que, sumado a las condiciones de trabajo como son temperatura, humedad, y partículas en el ambiente, hacen que los elementos tiendan a fallar con el transcurso del tiempo.

A continuación damos un breve repaso a los fallos que podemos encontrar en los elementos (engranes y rodamientos) que forman nuestros sistemas de estudio.

2.3.1. Fallo en engranes

Los engranes usados en maquinaria industrial son manufacturados en materiales altamente resistentes a la corrosión, desgaste, fatiga y esfuerzos de deformación con el afán de soportar condiciones severas de funcionamiento. Sin embargo, defectos en el material, mala fabricación o una deficiente lubricación, pueden generar un fallo prematuro.

Tal y como muestra la Figura 2.4, las principales causas de fallo en engranes se deben principalmente a circunstancias directamente relacionadas con el proceso de lubricación, por lo que la parte más afectada en estos elementos serán las superficies de contacto entre dientes.

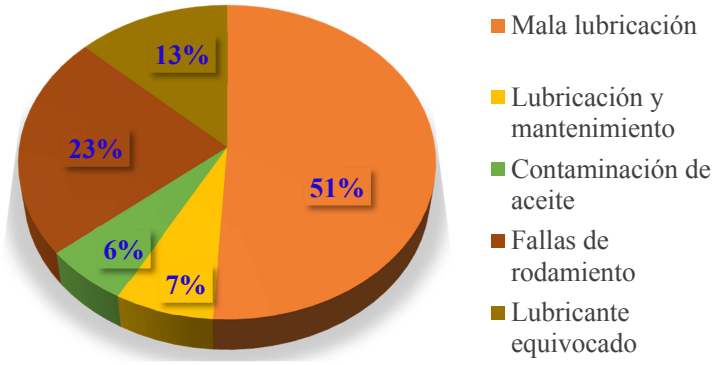


Figura 2.4: Principales causas de fallos en engranes [1].

Una clasificación detallada de los fallos que pueden ocurrir en engranes por diferentes causas es mostrada a continuación:

- **Perturbaciones superficiales:** Las perturbaciones superficiales son fallos que se presentan en la cara del diente en engranes que se encuentran sometidos a condiciones adversas de funcionamiento (como pueden ser una mala lubricación, presencia de sustancias extrañas en el lubricante, descargas eléctricas, o excesiva carga). Estas perturbaciones se pueden clasificar en varias familias según su origen y tipología:

- Desgaste por deslizamiento: desgaste normal, desgaste moderado, desgaste excesivo, pulido, desgaste abrasivo, rascado moderado, rascado severo, desgaste de interferencia.
- Corrosión: corrosión química, corrosión por fricción, decapado.
- Sobrecalentamiento.
- Erosión: cavitación, hidráulica, eléctrica.

En general, la detección temprana de este tipo de fallos evitará problemas futuros, pero un engrane con ellos podrá seguir en funcionamiento durante largos periodos de tiempo.

- **Raspaduras:** Las raspaduras son fallos generados por fatiga superficial entre dientes de engranes, y se presenta incluso con adecuada lubricación. Dependiendo del tiempo transcurrido desde su aparición, este fallo puede presentarse bajo tres niveles de severidad: leve, moderada, y severa.
- **Deformaciones permanentes:** El fallo por deformaciones permanentes se presenta en engranes una vez superada la resistencia última del material causado por una carga externa una vez que es retirada. Este tipo de fallo puede ser prevenido con el uso de grasas para presión extrema. Una posible clasificación de este tipo de fallos es:

  - Abolladura.
  - Deformación plástica: por laminado, por contacto.
  - Ondulación.
  - Deformación por arista viva.
  - Deformación por rebaba.

- **Fenómeno de fatiga superficial:** El fenómeno de fatiga superficial se presenta en engranes con capa delgada de lubricación o por el uso de un aceite inadecuado, y que además se encuentran sometidos a tensiones elevadas de trabajo, lo que provoca pérdida de material en la cara de los dientes. Este tipo de fallo se subdivide en seis posibles categorías:

  - Picaduras: picaduras iniciales, picaduras progresivas, macro-picaduras.
  - Picaduras escamosas.
  - Astillamiento.
  - Trituración.

  Las picaduras iniciales se presentan en la mayor parte de engranes en etapas iniciales de uso, lo cual no representa problemas para su funcionamiento. Sin embargo, el panorama cambia cuando se trata de picaduras progresivas, que acaban con la superficie del diente y provocan ruido durante su funcionamiento. Las picaduras progresivas comienzan como pequeños agujeros bajo el diámetro de paso del engrane y a medida que avanza cubre toda la superficie del diente.

- **Roturas y fisuras:** Las fisuras son el resultado de altas tensiones mecánicas, temperaturas de trabajo elevadas, o un mal tratamiento térmico empleado durante la manufactura del engrane. Este fallo, de ser detectado en etapas tempranas, no supondrá averías del sistema, pero si no se trata de forma adecuada avanzará llegando a generar roturas de diente que desgastarán rápidamente al sistema y provocará una disminución abrupta en su vida útil. Este fallo puede clasificarse en:
  - Grietas de temple.
  - Grietas de rectificado.
  - Grietas por fatiga.
- **Diente roto:** El fallo por diente roto genera problemas de operación en la caja de engranes debido a la exigencia que ocasionará en los dientes que aún están en buen estado, pudiendo provocar el fracaso de todo el sistema. El fallo dependerá de factores externos como la lubricación, la carga a la cual están sometidos, e incluso su proceso de manufactura. A continuación se presenta una posible clasificación de este tipo de fallos:
  - Rotura por sobrecarga: fractura frágil, fractura dúctil, fractura semi-frágil.
  - Cizallado de diente.
  - Rotura posterior a deformación plástica.
  - Rotura por fatiga: fatiga de dientes, rotura de dientes.

2.3.2. Fallos en rodamientos

Debido a las distintas condiciones bajo las que puede trabajar un rodamiento, cada uno de los elementos que lo componen puede fallar con el paso del tiempo de funcionamiento. Al igual que ocurría en el caso anterior, la causa principal de fallos en rodamientos se presenta por un defecto en el lubricante, ya sea por envejecimiento (horas de funcionamiento) o por una mala elección del mismo (viscosidad, gradiente de temperatura, etc), lo que puede provocar que los elementos del rodamiento se deterioren y no funcionen de una manera adecuada, llegando incluso a generar graves daños al equipo y pérdidas económicas considerables a causa del tiempo requerido para realizar un cambio del mecanismo. A continuación se muestran los distintos tipos de fallos que pueden aparecen en los rodamientos:

- **Fatiga por contacto:** El movimiento continuo y giratorio al cual están sometidos los rodamientos durante su funcionamiento hacen que la incidencia del fallo de fatiga por contacto sea elevada incluso en condiciones normales de carga y velocidad. Este fallo se subdivide en dos categorías:
  - Fatiga inicial bajo la superficie.
  - Fatiga superficial.

- **Desgaste:** Durante el funcionamiento de un rodamiento existen situaciones que provocan una pérdida del material de alguno de sus elementos, que se denomina desgaste. Este fallo también es causado principalmente por circunstancias asociadas al lubricante y requiere especial cuidado para evitar problemas futuros. Dependiendo de la causa externa que lo provoca el desgaste se puede clasificar en:
  - Desgaste abrasivo.
  - Desgaste adhesivo.
  - Desgaste por contacto: desgaste corrosivo, desgaste por partículas.
- **Corrosión:** El fallo por corrosión tiene características similares al desgaste. Su única diferencia es la causa que originó el problema. De acuerdo con esto, se clasifica en:
  - Corrosión por humedad.
  - Corrosión friccionante: corrosión friccionante leve, corrosión friccionante severa.
- **Erosión eléctrica:** El fallo por erosión eléctrica es un problema recurrente al utilizar variadores de frecuencia. Este fallo se debe a la corrosión producida por fugas de corriente o voltajes elevados, y se produce al pasar corriente por los elementos del rodamiento generando calor y, con ello, corrugaciones en las pistas, de tal forma que el lubricante queda inservible y los elementos con daños permanentes. El incremento que se ha dado a partir del año de 1990 en el uso y desarrollo de variadores de velocidad para motores, hacen que este fallo se presente con alta frecuencia a nivel industrial.
- **Deformación plástica:** Tras cumplir con su tiempo de vida útil, los rodamientos comienzan a experimentar fallos por deformación plástica que se presentan como abolladuras en la superficie tanto de la pista externa como interna. Este proceso normal de fallo en rodamientos puede aparecer antes de cumplir con su vida útil cuando el elemento está expuesto a cargas extremas que superen su diseño, excesivas pre-cargas, o cargas axiales. Se puede clasificar como:
  - Deformaciones por sobrecarga.
  - Abolladura: abolladura por escombros, abolladura por manipulación.
- **Grietas y fracturas:** El fallo por grietas y fracturas se presenta en rodamientos montados de manera inadecuada, fuerzas externas, mala lubricación, o sometidos a cargas elevadas. En etapas iniciales de fallo se presenta en forma de grietas, las cuales generarán ruido durante su funcionamiento. Al pasar el tiempo la grieta terminará en una fractura del elemento, lo que causará una parada abrupta del mecanismo del cual forma parte y posibles daños adicionales. Su clasificación de acuerdo al origen del fallo es:
  - Fractura forzada.

- Fractura por fatiga.
- Grieta por temperatura.

2.4. Análisis Tiempo-Frecuencia

El análisis de Fourier permite obtener información clara sobre las componentes de frecuencia que tiene una señal. Este análisis es útil, principalmente, cuando la señal es periódica, siendo posible representar una serie de tiempo como una suma de múltiples componentes senoidales y cosenoidales, lo que permite expresar la transformada de Fourier de forma más compacta haciendo uso de la exponencial compleja.

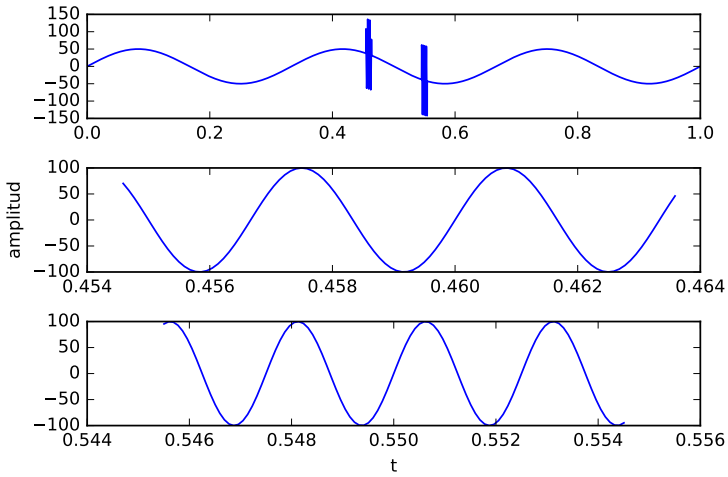
Sin embargo, no todas las señales cumplen una condición de periodicidad, algunas presentan eventos que ocurren una sola vez durante todo el dominio de la señal capturada y otras responden a procesos subyacentes que evolucionan en el tiempo. En estos casos, el análisis de Fourier más clásico no es capaz de extraer patrones de repetición locales, por lo que proporciona una herramienta limitada para múltiples aplicaciones que requieren del procesamiento de este tipo de series temporales.

Por ejemplo, la Figura 2.5 muestra una señal de $3Hz$ (Figura 2.5a superior) en la que se han inyectado dos componentes de $300Hz$ y $400Hz$ respectivamente (Figura 2.5a inferior). En la Figura 2.5b se muestra el espectro de la señal, en la que es apreciable la pérdida de información. Si bien es cierto que existen picos de frecuencia cerca de los $300Hz$ y $400Hz$, indicando la presencia de los transitorios, estas bandas frecuenciales tienden a traslaparse y por tanto a confundirse. Es apreciable, además, que la información concerniente a la baja frecuencia de $3Hz$ ha desaparecido por completo. Además, al ser eventos que ocurren en un tiempo específico y una sola vez, sería interesante poder conocer en qué instante de tiempo sucedieron, pero esta información se encuentra completamente fuera del alcance de la transformada de Fourier, que es capaz de decirnos qué frecuencias ocurren, pero no cuándo ocurren.

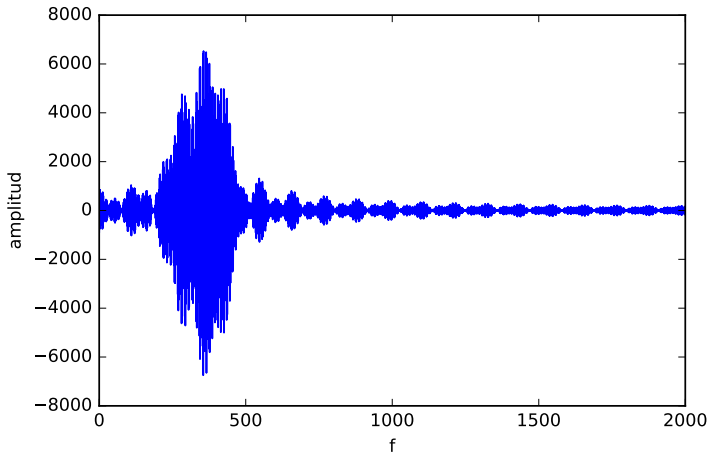
2.4.1. Transformada Corta de Fourier

Debido a las limitaciones indicadas anteriormente ha sido necesario el desarrollo de otro tipo de herramientas para el análisis de señales. Éstas debían ser capaces de extraer patrones, tanto globales como locales, de las series temporales. Normalmente, los patrones globales se presentan en las señales como componentes de baja frecuencia con una tasa de ocurrencia baja, mientras que los patrones locales suelen aparecer como eventos de corta duración (alta frecuencia) y de forma esporádica, incluso única, como una parte transitoria de la señal capturada.

Es así como nace el análisis tiempo-frecuencia con una primera propuesta de Dennis Gabor en su artículo *"Theory of communication"*[30], donde se bautizó como *análisis por ventaneo de señales* y que actualmente se conoce como *Transformada Corta*



(a) Señal



(b) Espectro

Figura 2.5: Señal con distorsión de transitorios de alta frecuencia y su espectro.

de Fourier (STFT). Este proceso dio una primera solución para el descubrimiento de patrones locales en series temporales, permitiendo conocer un rango de tiempo en el cual puede haber ocurrido un evento de alta frecuencia.

La STFT se realiza aplicando la transformada de Fourier a trozos sucesivos y ordenados de la señal. Estos trozos son determinados por el tamaño de una ventana previamente definida de acuerdo a la aplicación objetivo. Con la STFT se obtiene el espectro de cada tramo de tiempo, logrando conocer de esta forma las componentes de frecuencia para cada uno de ellos. Así, es posible saber que un evento de alta frecuencia ha ocurrido en un tramo de tiempo si éste aparece en la transformada de Fourier específica para ese tramo de tiempo. La Figura 2.6 muestra la STFT para la señal de la Figura 2.5a, donde se pone de manifiesto su capacidad para encontrar las componentes de alta frecuencia que ocurren en rangos de tiempo específicos. Además, es posible notar que la componente de baja frecuencia permanece inalterada y visible en todo instante de tiempo, evitando la pérdida de información.

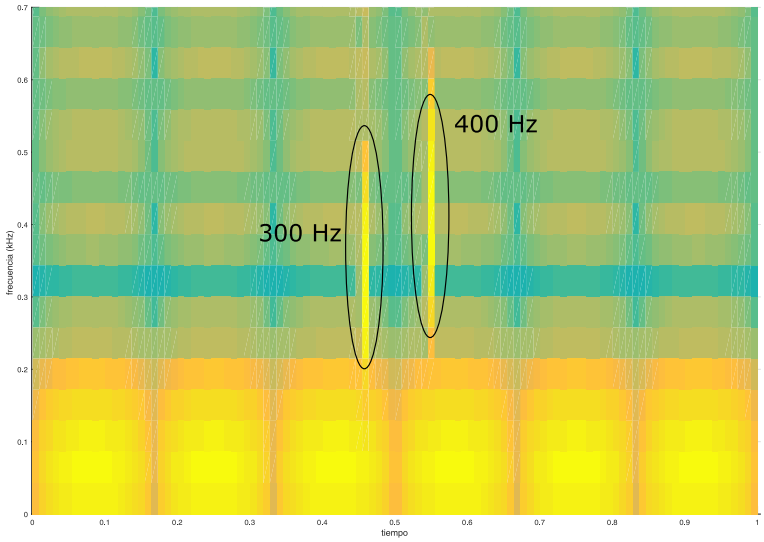


Figura 2.6: Ejemplo de STFT.

2.4.2. Transformada Wavelet

La Transformada Corta de Fourier ha sido una herramienta muy útil para el procesamiento de señales, sin embargo tiene dos deficiencias claves. La primera es que para cada posición de la ventana se calcula la transformada de Fourier, lo que resulta en un procedimiento computacionalmente muy costoso y que además depende del tamaño de la ventana seleccionada, sobre todo para aplicaciones donde se requiere una buena resolución en tiempo (ventanas angostas) para poder detectar eventos de alta frecuencia. La segunda de las deficiencias es que el ancho de la ventana es fijo a lo largo de todo el procedimiento, lo que produce una resolución fija tanto en tiempo

como en frecuencia, a pesar de ser conocido que los eventos de alta frecuencia necesitan mayor resolución en tiempo y los de baja frecuencia precisan mayor resolución en frecuencia.

Debido a las deficiencias de la transformada de Fourier y sus derivaciones, en los años 80 empiezan a sentarse las bases de una nueva teoría para el procesamiento de señales, a la que llamaron *Análisis de Wavelet* que, aunque no fue el primer intento de abordar la representación tiempo-frecuencia desde esta perspectiva, sí fue el más fructífero. Su desarrollo se debió principalmente a los aportes de Stephane Mallat en su trabajo *"A theory for multiresolution signal decomposition: the wavelet representation"*[63], donde se presentaban algoritmos para el cálculo eficiente del análisis de Wavelet, y al trabajo *"Orthonormal bases of compactly supported wavelets"*[24] de Ingrid Daubechies, donde se propone un conjunto de bases ortonormales que definen wavelets útiles para múltiples aplicaciones.

Una wavelet es una señal de corta duración con energía finita, localizada en un intervalo de tiempo específico. Un ejemplo de wavelet se muestra en la Figura 2.7.

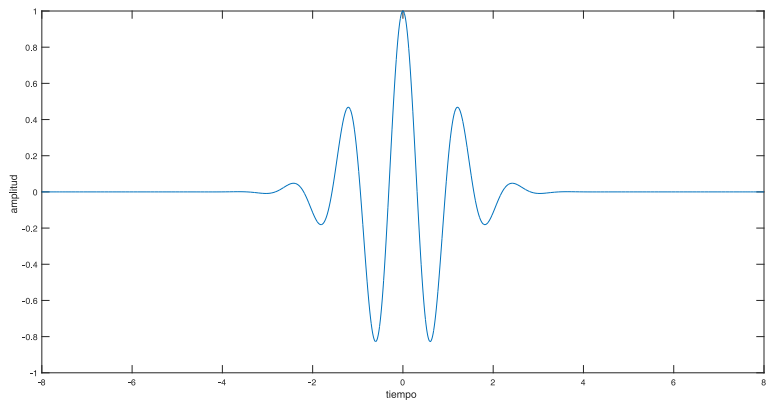


Figura 2.7: Ejemplo de wavelet.

El Análisis de Wavelet, al igual que el de Fourier, está basado en la descomposición de la señal en un conjunto de componentes. Sin embargo, su principal diferencia radica en que en este caso las funciones base utilizadas, también conocidas como *wavelet madre*, tienen una duración finita y, además de cambiar su frecuencia (como en el análisis clásico de Fourier) también cambian su posición temporal. De esta forma, la transformada wavelet permite ajustar la resolución tanto en tiempo como en frecuencia, obteniendo una mejor resolución en tiempo para eventos de alta frecuencia y una mejor resolución en frecuencia para las componentes de baja frecuencia. Es necesario tener en cuenta que, debido al *Principio de Incertidumbre de Heisenberg*[34], traspasando un umbral no se puede tener una mejora en la resolución en tiempo sin empeorar la resolución en frecuencia y viceversa.

Tomando en cuenta lo dicho, la *Transformada Discreta de Wavelet* (DWT) utilizada

en señales discretas viene dada por:

$$c(j,k) = \sum_{n\in\mathbb{Z}} x(n)\psi_{j,k}(n) \tag{2.7}$$

donde $\psi_{j,k}$ es un elemento de la familia de wavelets definida por la base ψ que viene dado por:

$$\psi_{j,k}(n) = 2^{-\frac{j}{2}} \cdot \psi(2^{-j}n - k) \tag{2.8}$$

Como se mencionó anteriormente, la diferencia entre la transformada de Wavelet y la de Fourier radica en el hecho de que en la primera la función base es sometida a un cambio tanto en frecuencia como posición en el tiempo, algo que se encuentra implícito en la ecuación (2.7), donde el parámetro de traslación, τ, y el de escala, s (el recíproco de la frecuencia), vienen determinados por:

$$\tau = 2^j \tag{2.9}$$

$$s = 2^j k \tag{2.10}$$

2.4.3. Análisis multi-resolución

El *Análisis Multi-resolución* (MRA), conocido en un principio como *Algoritmo Piramidal*, fue creado en 1983 por Peter J. Burt para la representación compacta de imágenes [12]. Posteriormente, en 1989 Mallat encontró una relación entre este algoritmo, la teoría de filtros, y el Análisis de Wavelet [63]. A partir de estos resultados se crea un conjunto de algoritmos basados en el MRA para la descomposición de señales. El fin inmediato era disponer de un conjunto de herramientas para calcular la DWT, equivalentes a la FFT con respecto a la disminución en el coste computacional, y encontrar una representación tiempo-frecuencia obtenida de forma adaptativa con respecto a su resolución en los dos dominios.

El cálculo de la DWT basado en el MRA establece que una señal puede ser descompuesta en sus componentes de alta frecuencia, d, y baja frecuencia, a, mediante la aplicación de filtros con respuesta a los impulsos $h(n)$ y $g(n)$, respectivamente, seguido de una operación de submuestreo con factor 2. Esto es válido siempre y cuando los filtros se encuentren relacionados por la ecuación

$$g(L-1-n) = (-1)^n \cdot h(n)$$

lo que determina que se llamen *Filtros Espejo en Cuadratura* (QMF).

Esta relación se puede formalizar como:

$$d(k) = \sum_n x(n) \cdot g(2k-n) \tag{2.11}$$

$$a(k) = \sum_n x(n) \cdot h(2k-n) \tag{2.12}$$

Este proceso de descomposición se aplica de forma recursiva sobre la componente de alta frecuencia, a, que comúnmente se denomina *señal de coeficientes de aproximación*, de tal forma que se efectúa la DWT mediante MRA. Con cada aplicación sucesiva de la ecuación (2.11) y la ecuación (2.12) se construye un nuevo nivel de descomposición. Este proceso puede continuar hasta que la componente a tenga un único elemento. Sin embargo, en la práctica el nivel de descomposición se limita con algún criterio de optimización, o bien determinado por la aplicación específica. Si la ecuación (2.11) y la ecuación (2.12) se aplican también a d entonces el nombre que se le da es el de *Transformada de Wavelet Packet* (WPT)[31], lo que resulta en una descomposición óptima de la señal para obtener una representación tiempo-frecuencia.

Un ejemplo de la WPT aplicada a la señal que hemos venido estudiando para obtener su representación tiempo-frecuencia se muestra en la Figura 2.8, donde se puede observar que el ratio tiempo-frecuencia se encuentra mejor distribuido que en el análisis equivalente que se realizó con STFT.

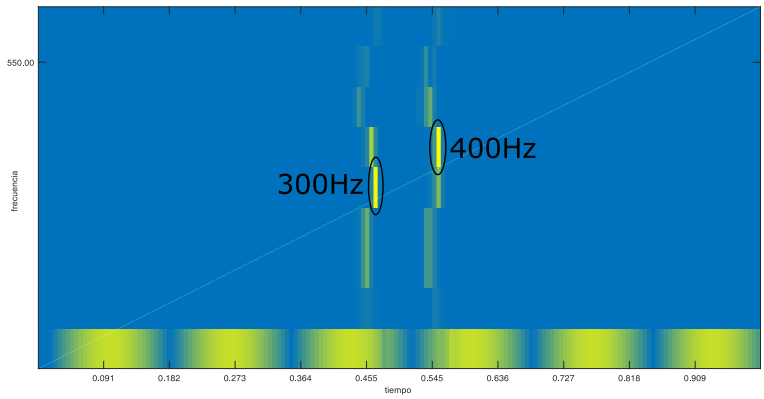


Figura 2.8: Ejemplo de espectrograma con WPT.

2.5. Sistema de adquisición de datos

Antes de la tarea de analizar series de tiempo que corresponden con mediciones reales de sistemas mecánicos se encuentra un proceso no trivial de captura de datos que merece ser estudiado con mayor profundidad.

Para empezar, definiremos un *sistema de adquisición de datos* como un conjunto ordenado de elementos y componentes físicos relacionados entre sí de manera que en grupo forman una unidad completa y que pueden actuar como tal, ya sea de manera secuencial o simultánea, para obtener datos de las variables de un proceso concreto.

En los inicios de la adquisición de datos, el proceso de captura de datos de una variable se realizaba de forma manual mediante inspección de las lecturas obtenidas en instrumentos analógicos de medición. Los mecanismos de estos instrumen-

tos eran capaces de reaccionar ante cambios de la variable física para los que fueron diseñados, y provocaban el movimiento de un elemento de visualización, habitualmente de forma proporcional a los cambios percibidos en la variable observada (por ejemplo, un manómetro para la medición de la presión del aire). Las lecturas eran almacenadas en registros escritos para posteriormente ser analizados, representados en gráficas, o simplemente almacenados.

Sin embargo, esta metodología presenta varios inconvenientes, por ejemplo: los errores de medición por apreciación, o por fallos en la calibración de los instrumentos, afectan a la exactitud de la medida; el ruido en la medición afecta a la precisión de los datos; el tiempo de captura dato a dato puede ser demasiado largo y costoso; no es posible sincronizar las mediciones de múltiples variables, y además no es posible obtener datos con un intervalo constante (baja latencia). Parte de estos inconvenientes fueron minimizados mediante el reemplazo de los elementos mecánicos por transductores que son capaces de transformar la variable observada en un potencial eléctrico que puede ser medido. Sin embargo, a pesar de los esfuerzos en mejorar la tecnología de medición de las variables, no era posible reducir el tiempo de captura, obtener una baja latencia, o sincronizar la captura de múltiples variables.

Es así como, tras múltiples intentos dispersos realizados por varias compañías tecnológicas, la división de procesamiento de datos de IBM[2] saca al mercado en 1963 el IBM® 7700®, convirtiéndose en el primer sistema de adquisición de datos comercial. Este era un sistema de 18 bits capaz de realizar operaciones aritméticas simples en 2 ciclos de máquina, donde cada ciclo era ejecutado en 2 microsegundos. Además, era capaz de recolectar datos de forma simultánea desde 32 fuentes, procesarlos y mostrar los resultados como un documento impreso o de forma visual en hasta 16 terminales diferentes. En 1964, el IBM® 7700® sería reemplazado por el IBM® 1800®, que tenía la capacidad de trabajar con cientos de variables de proceso de cualquier tipo, lo que lo convertía en un sistema de adquisición de datos de propósito general. Todos estos sistemas eran productos aislados sin compatibilidad con otros productos de cómputo, de precio muy elevado, y por ende solamente accesibles a un sector industrial reducido. En 1981, la empresa Scientific Solutions INC crea la línea LABMASTER® de productos para la adquisición de datos, compatible con los ordenadores personales (PC) de IBM, con lo que se marca el inicio de la era de los sistemas de adquisición de datos basados en PC, que perdura hasta la actualidad.

La Figura 2.9 muestra un diagrama general de un proceso de adquisición de señales (la evolución temporal de las variables observadas), donde se aprecia que la señal correspondiente al proceso físico es capturada por medio de sensores especializados acorde a cada variable medida. Estos sensores miden las señales físicas y las convierten en señales eléctricas que son enviadas a una etapa de acondicionamiento de la señal, con la finalidad de eliminar el ruido eléctrico que podría afectar la medición y amplificar los niveles de la señal capturada en la etapa anterior. Tras el proceso de acondicionamiento, los datos, que hasta el momento no son más que

[2]Siglas de la empresa International Business Machines Corporation.

niveles de voltaje continuo, son convertidos a datos digitales por medio de un proceso de conversión analógico-digital. Luego, estos datos digitales son enviados al computador para su almacenamiento, procesamiento y análisis.

A continuación vamos a detallar cada etapa del sistema de adquisición de señales aquí mostrado. Hemos obviado la etapa **Proceso o Planta**, mostrado en el diagrama, debido a que no forma parte del sistema de adquisición, sino que representa la fuente de donde se obtienen los datos.

- **Sensores y Transductores**: El *sensor* es el elemento que está directamente relacionado con la magnitud o variable física que se desea medir. Un sensor está constituido por un material o mecanismo complejo que altera sus propiedades en función del cambio en la variable física de interés. Por ejemplo, un sensor de temperatura típicamente cambia su resistencia eléctrica en función de la temperatura a la que está sometido, y esta variación puede ser captada por medidores de resistencia eléctrica, llamados Ohmetros.

  Sin embargo, en la mayoría de los sistemas de adquisición de datos es necesario trabajar con señales eléctricas de voltaje o de corriente, y es aquí donde interviene el *transductor*, un elemento que transforma la variación del sensor, provocada por la variable física, en una señal de tipo eléctrica. Tomando como ejemplo el sensor de temperatura antes mencionado, junto al sensor será necesario un transductor que tome la variación de la resistencia y la transforme en un cambio en la corriente eléctrica o voltaje. Lo deseado en la variación, tanto del sensor como en el transductor, es que sea proporcional a la variación de la variable física, es decir, a un cambio constante, Δ_f, de la variable medida debe corresponder un cambio constante, Δ_s, en la medida eléctrica resultante del transductor. Sin embargo, esto no es posible en todo tipo de variables, por lo que podemos encontrar sensores que tienen un comportamiento lineal por intervalos y otros cuyo comportamiento es enteramente no-lineal ante la variación en el proceso.

- **Acondicionamiento de la Señal**: La señal eléctrica entregada por el transductor, ya sea de voltaje o de corriente, normalmente se encuentra contaminada con el llamado *ruido eléctrico*. El ruido eléctrico en una medición es el resultado de la adición, a esa medición, de otras señales que se encuentran en el mismo entorno o que se han filtrado utilizando algún medio de propagación, como por ejemplo el aire. Este fenómeno resulta perjudicial para el proceso de adquisición porque distorsiona la señal original, dando como resultado información errónea. La afección puede variar dependiendo del ratio entre la magnitud de la señal original y la magnitud del ruido. Si este ratio es grande, la afección es pequeña, pero si es muy pequeño significa que el ruido opaca la mayoría de la información de la señal original y por lo tanto la afección es mucho mayor. Un ejemplo de ruido eléctrico es la componente de $50 - 60Hz$ (dependiendo del país) que suele aparecer en algunas mediciones a causa del suministro de la red eléctrica.

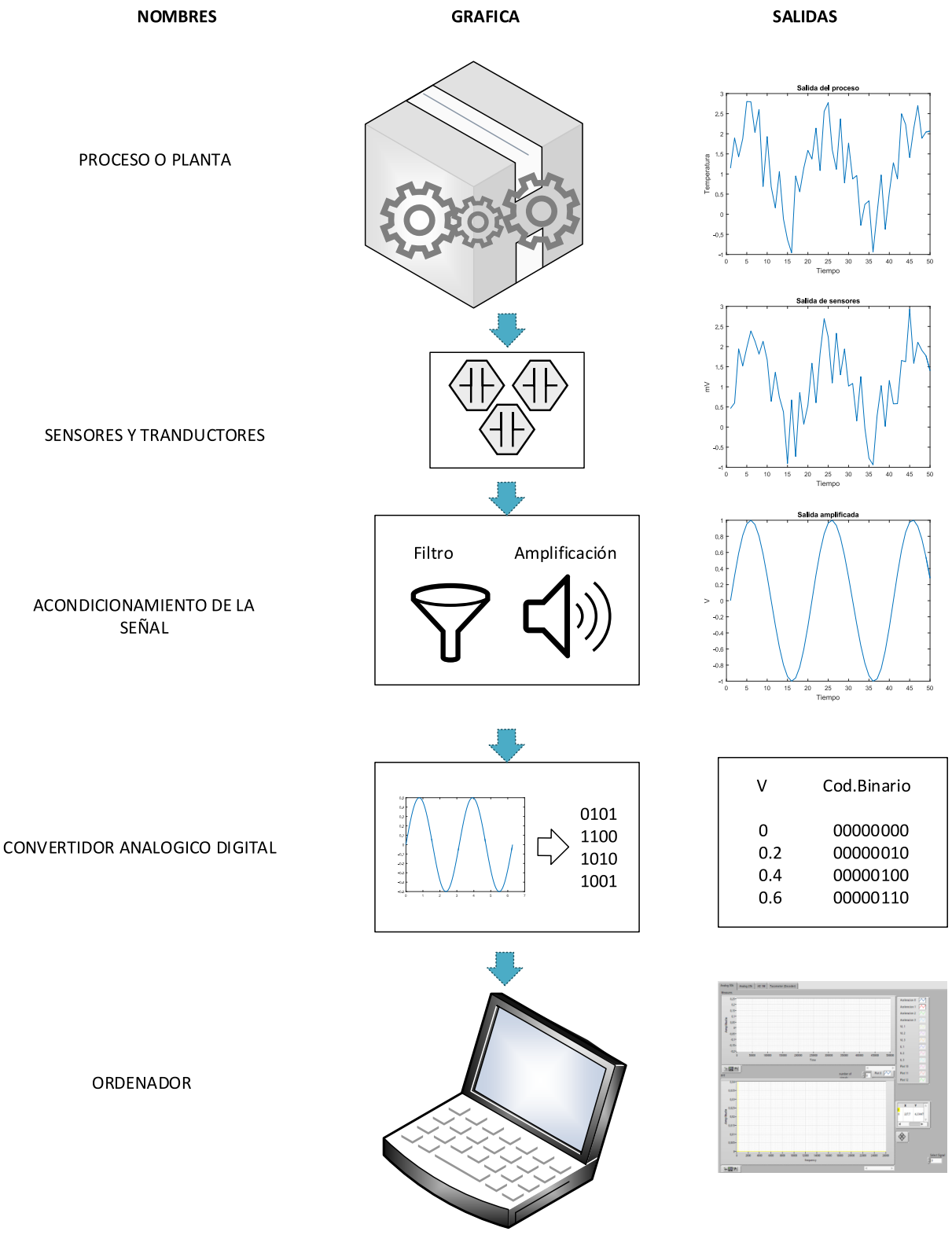


Figura 2.9: Diagrama de un sistema de adquisición de señales.

Una de las funciones del proceso de acondicionamiento de la señal es la de eliminar el ruido eléctrico de la señal original o, al menos, de mitigar su efecto en caso de que la eliminación completa no sea posible. Para ello, se realiza un proceso de filtrado que elimina las componentes de frecuencia específica, que a priori se sepa que no son útiles, mediante filtros *elimina banda*. Otra forma de resolver el problema es hacer uso de filtros *pasa banda* para eliminar todas las componentes de frecuencia que no se encuentren en el rango frecuencial en el cual la variable proporciona información útil. Estos dos métodos no son excluyentes, y pueden ser usados en una misma etapa de acondicionamiento de la señal incluso combinados con otro tipo de filtros como los *pasa bajo* y *pasa alto*. Todo dependerá del tipo de ruido que se desee eliminar y de la importancia de disponer de una señal pura para la aplicación posterior a la adquisición de la señal.

Otra función de esta etapa es la amplificación de la señal original. En muchos casos la señal que se recibe tiene un nivel de amplitud muy pequeño, en el orden de los mili o microvoltios (por ejemplo, las señales que se adquieren a partir de mediciones en los músculos del cuerpo humano). Para estos casos, se captura la señal original y se intenta amplificar de forma controlada conservando las propiedades de forma y fase que tenía originalmente y evitando la inclusión de retardos en la señal resultante. Muchas veces, este proceso se realiza mediante amplificadores analógicos que en muchos casos podrían ser parte de los circuitos de filtrado detallados anteriormente, realizando a la vez la doble función de filtro y amplificador.

- **Conversión analógica a digital**: En la actualidad, las técnicas y algoritmos de análisis y almacenamiento de señales están implementados en dispositivos de computación que, como vimos, forman parte fundamental de los sistemas de adquisición de señales. Estos dispositivos están formados por circuitos electrónicos que son capaces de trabajar con información que se encuentra codificada en secuencias de bits. Por ello, para que puedan procesar o almacenar señales, éstas deben estar igualmente representadas como secuencias de bits. Sin embargo, las señales de salida de la etapa de acondicionamiento son analógicas y continuas, por lo que en un sistema de adquisición de señales es necesario disponer de una etapa dedicada a la conversión de señales analógicas a su codificación digital equivalente. Un diagrama del proceso de conversión analógico a digital se muestra en la Figura 2.10.

  El proceso de conversión analógico a digital (ADC) empieza con una etapa de muestreo y retención. Como el proceso de conversión tarda un periodo de tiempo, aquí se captura un nivel de voltaje de la señal con el circuito de muestreo y se mantiene fijo hasta que se termine la conversión con el circuito de retención, algo necesario para evitar que cambie el nivel de voltaje del cual queremos conocer su codificación digital. Una vez que la señal se fija como un voltaje de referencia (muestreo y retención) empieza la etapa de conversión analógica a digital. Para ello se utiliza un circuito de conversión digital a analógico (DAC) con un código digital ascendente, que da un valor de voltaje a

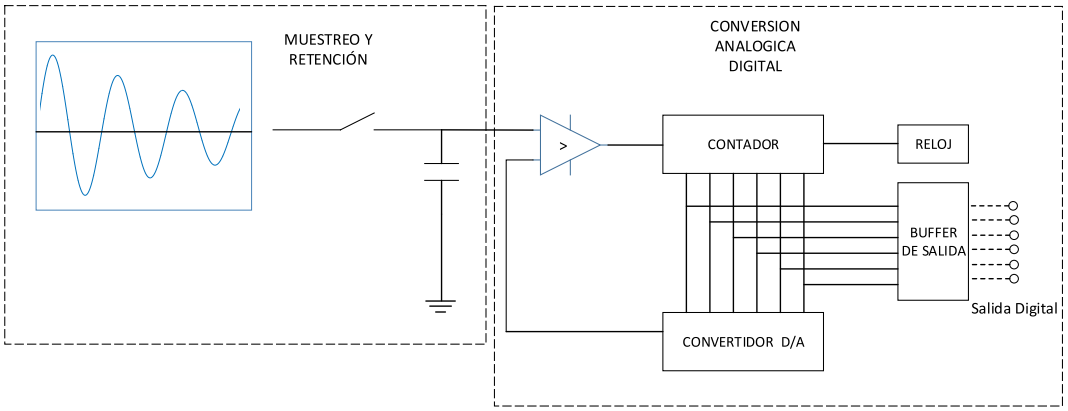


Figura 2.10: Diagrama del conversor analógico digital.

un comparador que dispara un impulso de parada cuando el voltaje del DAC es igual o mayor que la referencia dada por el circuito de muestreo y retención. Cuando el impulso de parada se activa, se detiene la cuenta ascendente del código digital en el DAC y se toma este valor binario como el código que representa el nivel de voltaje que se mantuvo retenido. Este valor es el resultado de la ADC para un periodo de tiempo. Posteriormente, se muestrea de nuevo y se retiene otro nivel de voltaje y el proceso se repite.

Para la selección de los dispositivos electrónicos que realizan la conversión analógica a digital se deben tener en cuenta al menos dos factores fundamentales: la tasa de muestreo y la resolución. La primera representa el número de ADCs que se pueden realizar en un tiempo determinado, y habitualmente de mide en $samples/s$. La segunda representa la variación mínima de voltaje que puede ser detectada por el dispositivo de ADC y que por lo tanto cambia el valor binario resultante. Para entender las restricciones que imponen estos dos parámetros de forma práctica se puede decir que la tasa de muestreo se encuentra limitada en un caso extremo por el tiempo que tarda al DAC en realizar el barrido desde el código binario mínimo al máximo. Por otro lado la resolución se encuentra limitada por el número de bits que tiene el DAC.

- **Ordenador**: Todo el proceso realizado hasta este momento tiene la finalidad de transformar la señal, que inicialmente se encontraba como niveles de voltaje en el tiempo, en un conjunto de cadenas de bits que forman una representación discreta de la señal original. Esta versión digital suele ser enviada a un ordenador usando un medio cableado, aunque hoy en día también se utilizan medios inalámbricos. En cualquier caso, independientemente del medio usado para la transferencia de los datos, se usan protocolos de comunicación avanzados, que se encargan de gestionar la transferencia de la información y brindar robustez ante perturbaciones externas que pudiesen afectar al proceso de comunicación. En el lado del ordenador, un software de adquisición de datos es el encargado de gestionar la recepción de la señal digitalizada para

luego procesarla según las necesidades de la aplicación específica, donde se incluye su almacenamiento para su uso o análisis posterior. Si la aplicación es común en el medio industrial, se puede encontrar software comercial o incluso software libre para realizar esta tarea. Sin embargo, si se desea personalizar el software de adquisición con los requerimientos de una aplicación específica, entonces será necesario el diseño e implementación en algún lenguaje de programación o framework. Ejemplos de esto son LabView™, de la compañía National Instrument®, y Matlab[69], de la compañía MathWorks®.

2.5.1. Sistema de adquisición de datos para una máquina rotativa

Para este trabajo fue necesaria la implementación de un sistema de adquisición de datos que complementa a los sistemas mecánicos de la Figura 2.22, Figura 2.23, y Figura 2.25; y por el que se obtiene una observación de lo que pasa en una caja de engranes industrial o sistema de transmisión permitiendo además adquirir datos de manera que se puedan procesar y evaluar las señales obtenidas.

Parte del sistema de adquisición de datos está conformado por un subsistema eléctrico y electrónico que provee de energía a todos los componentes para su funcionamiento y además contiene toda la instrumentación para la medición de las variables. El esquema general de este subsistema se muestra en la Figura 2.11, donde se representan las conexiones eléctricas y electrónicas del banco de vibraciones. A continuación pasamos a detallar cada uno de sus componentes.

Sistema de alimentación de energía

En primer lugar describimos los elementos que proporcionan energía a los componentes del banco de vibraciones, es decir, aquellos que intervienen en el proceso que va desde la toma de energía de la red trifásica hasta el motor que produce el movimiento mecánico. El flujo energético se detalla en el diagrama de la Figura 2.12, donde se puede apreciar que la energía que alimenta al sistema se toma de una red trifásica para, posteriormente en un tablero de conexiones, ser acoplada por medio de fuentes de alimentación y ser distribuida según la necesidad de cada componente. A continuación, esta energía controlada será trasladada a los actuadores del sistema, que son el motor para dar el movimiento y el freno magnético para simular la carga mecánica.

La mayoría de componentes del sistema de alimentación de energía se encuentran en el tablero de conexiones. Un tablero de conexiones es una caja metálica con un conjunto de paneles internos que permiten realizar conexiones eléctricas y electrónicas aislando los elementos del medio en el que intervienen. Este tablero alberga diferentes componentes encargados de dar la respectiva energía a los sensores y de asegurarlos. Está provisto de un mando de emergencia que al ser presionado inhibe inmediatamente la energía y detiene cualquier proceso. Además, mantiene los componentes lejos del operario y es seguro para la distribución de energía hacia el

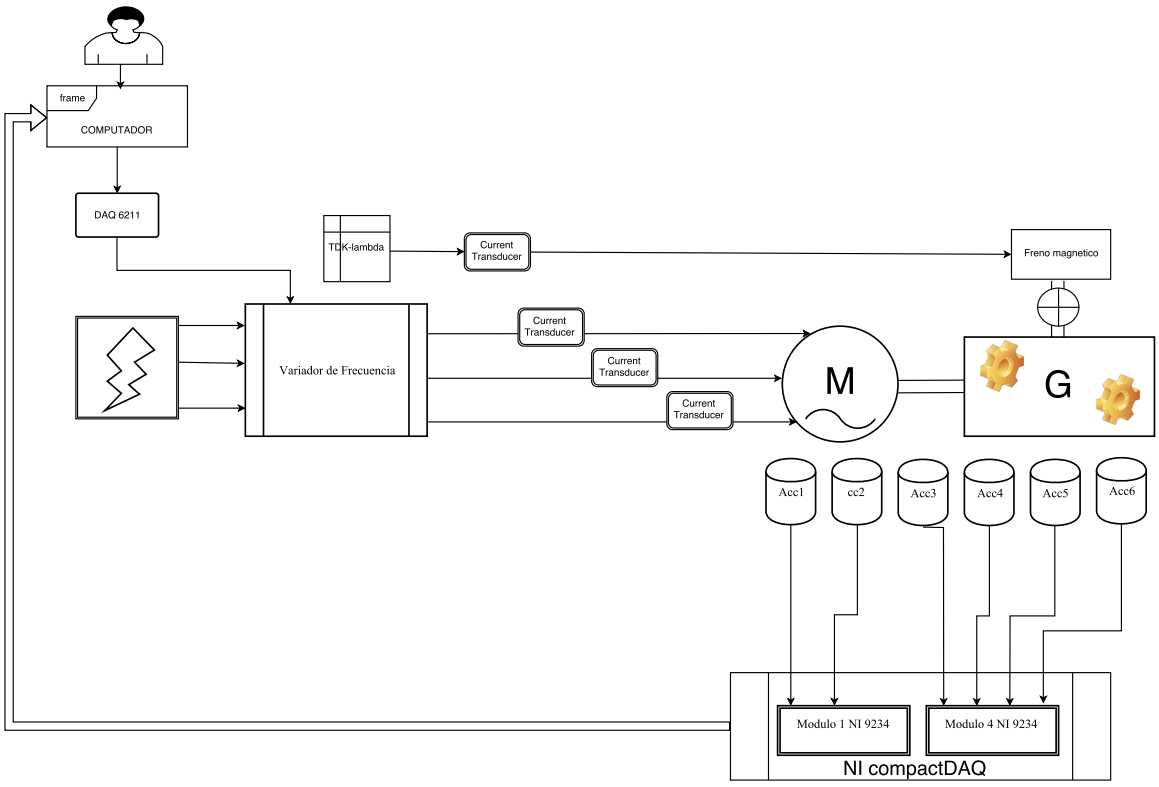


Figura 2.11: Diagrama eléctrico y electrónico de un sistema de adquisición de señales de vibración.

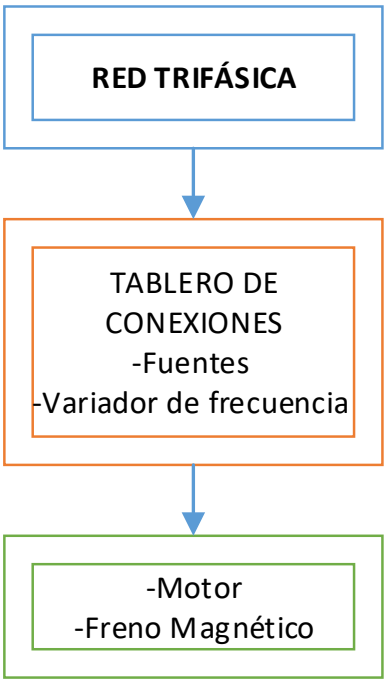


Figura 2.12: Diagrama eléctrico del sistema de adquisición de señales de vibración.

banco mecánico. Un tablero eléctrico y sus componentes para el banco de vibración se muestran en la Figura 2.13.

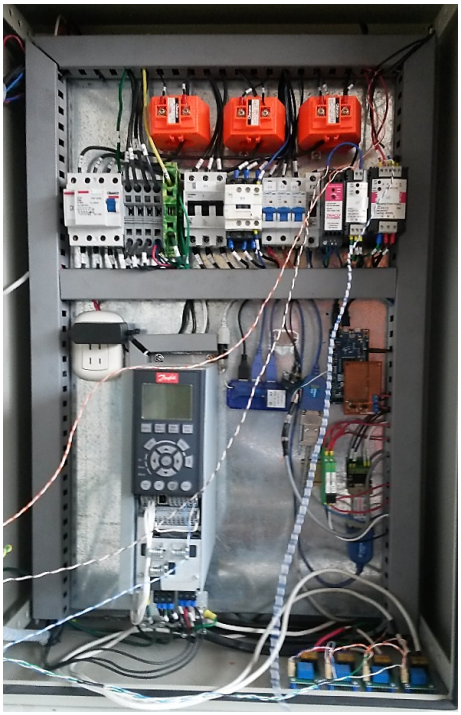


Figura 2.13: Tablero eléctrico principal.

Los elementos más importantes del tablero de conexiones y del motor son:

- **Medidor de corriente**: El medidor de corriente consiste en una bobina de corriente de montaje múltiple (Multi-mount current coil, en inglés) diseñada para obtener la medición de la corriente que circula por el cable que la atraviesa por el orificio entre sus electrodos (véase Figura 2.14). El medidor de corriente utilizado para el banco de vibraciones de las adquisiciones llevadas a cabo para este trabajo tiene las siguientes características:

  - Marca: Camsco.
  - Corriente: 60/ 5 A.
  - Frecuencia: 50 - 60Hz.

  El principio de funcionamiento de este sensor se basa en el efecto descubierto en 1879 por el físico Edwin Herbert Hall, y por el que fue llamado *efecto Hall*, que mostró que si se emplea un campo magnético elevado sobre una lámina conductora (normalmente, se usa una fina lámina de oro) por la que circula corriente, se produce un voltaje en una dirección transversal a la misma, este voltaje se llama *voltaje Hall*, que permite medir el voltaje de la lámina.

Figura 2.14: Sensor de corriente.

Este efecto es usado para medir la cantidad de corriente que pasa por un elemento conductor, y es el principio de funcionamiento de los sensores de efecto Hall usados en el proceso de mediciones del banco de vibraciones. Se usan tres de estos sensores para captar la cantidad de corriente que atraviesa cada línea de la red trifásica que ingresa en el tablero de conexiones.

- **Fuentes de alimentación industriales**: Las fuentes de alimentación industriales (Industrial Power Supplies) son las encargadas de suministrar los diferentes voltajes y corrientes a cada equipo. Están conectadas a la red trifásica de entrada. Para dar lugar a los diferentes voltajes se tienen tres fuentes, de 24V/2.5A, 12V/2.2A y 5V/4.0A, tal y como se muestra en la Figura 2.15.

  Las características de las fuentes de alimentación son:

  - Marca: Traco Power.
  - Voltajes de salida: 24V / 12V / 5V.
  - Corrientes de salida: 2.5A /0.5 -0.4 A /4.0A.
  - Modelo: TCL Q30-112 /TCL 060-124 / TCV 60-124.
  - Frecuencia: 50Hz/60Hz.
  - Potencia de salida: 60W / - /20W.

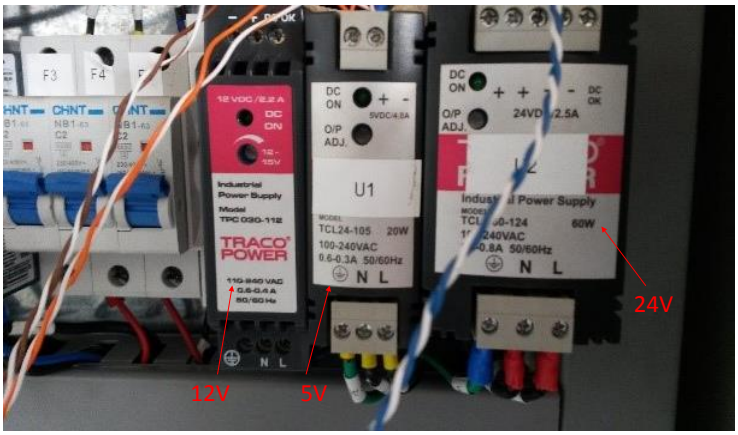


Figura 2.15: Fuentes de voltaje.

- **Variador de frecuencia**: Un variador de frecuencia es un dispositivo que se usa para controlar el motor con velocidades variables sin poner en riesgo la eficiencia del sistema. La eficiencia puede ser medida de manera rápida y global haciendo una relación entre los resultados obtenidos y la entrada total de energía al sistema:

$$EficienciaEnergetica = \frac{EnergiaEntregada}{EntradaTotalDeEnergia} \tag{2.13}$$

El funcionamiento de un variador de frecuencia consiste en tomar la energía, en este caso trifásica alterna, que tiene una magnitud y frecuencia fija, transformarla en un voltaje continuo para, posteriormente, transformarla nuevamente en voltaje alterno que tiene magnitud y frecuencia variable. El variador de frecuencia utilizado para el motor del banco de vibraciones se muestra en la Figura 2.16 y sus especificaciones aparecen a continuación:

- Marca: Danfoss.
- Modelo: FC 302 VLT® AutomationDrive.
- Corriente: 0.10 – 10000.0A.
- Voltaje: 10-1000V.
- Frecuencia: 0 – 1000Hz.

Figura 2.16: Variador de frecuencia.

- **Motor**: El motor trifásico es un dispositivo que transforma la energía eléctrica en energía mecánica gracias a la acción de campos magnéticos generados por bobinas. Estas máquinas eléctricas se componen de dos partes principales: el rotor, que realiza el movimiento, y el estator, que es la parte fija de la máquina. El motor utilizado es de marca Siemens, código 1LA7 090-4YA60 de 2 Hp de potencia, véase Figura 2.17.

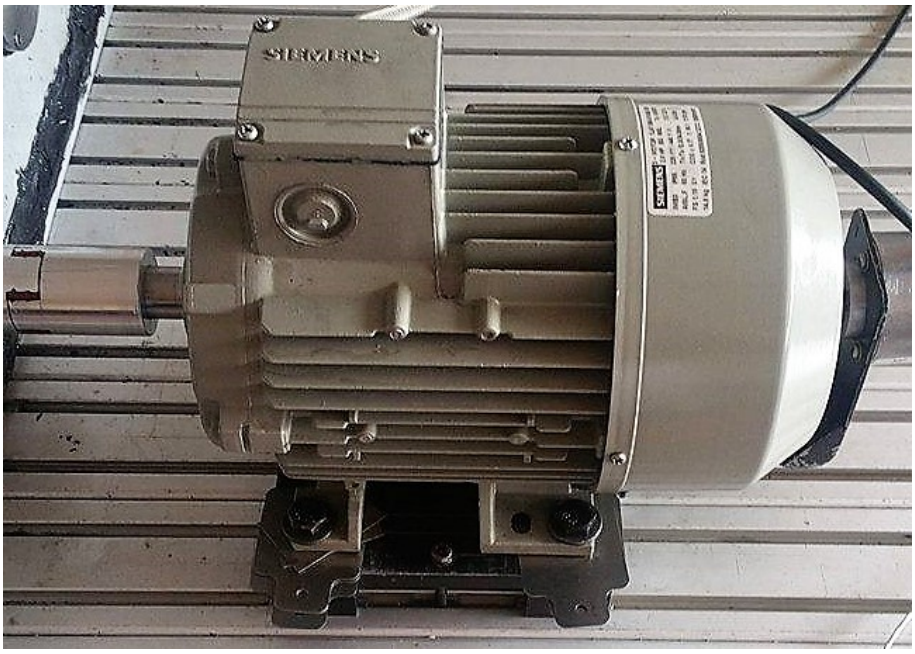


Figura 2.17: Motor eléctrico.

Sistema de instrumentación, adaptación y captura de señales

El sistema de instrumentación, adaptación y captura de señales está encabezado por el ordenador, que permite la interacción con el usuario, y es el encargado de enviar y recibir los datos necesarios para que se recolecten las diferentes señales de vibración del sistema. Un esquema de la interacción del ordenador con el resto de componentes se muestra en la Figura 2.18.

Un programa que se ejecuta en el ordenador es el encargado de controlar el variador de frecuencia que modifica la velocidad del motor conectado a la caja de engranes, y al mismo tiempo recibe los datos de las señales capturadas por los sensores. La comunicación entre el ordenador y el variador de frecuencia se lleva a cabo por medio de una tarjeta de adquisición de datos (DAQ) NI DAQ 6212, y el envío al ordenador de los datos obtenidos de los sensores se realiza usando un chasis NI CompactDAQ 9188.

Veamos algunos detalles de cada uno de estos componentes.

- **NI DAQ 6212**: Es un dispositivo de adquisición de datos que tiene la capacidad de medir y generar señales en cualquier momento. Para ello, se necesita un computador con el software adecuado instalado. Además, cuenta con entradas y salidas analógicas, entradas y salidas digitales, fuente de voltaje regulable de -15V/15V, y una fuente de voltaje fija de 5V.

  Se conecta con el computador mediante un cable USB y con los periféricos mediante cables apantallados para la supresión de ruido eléctrico exterior (en

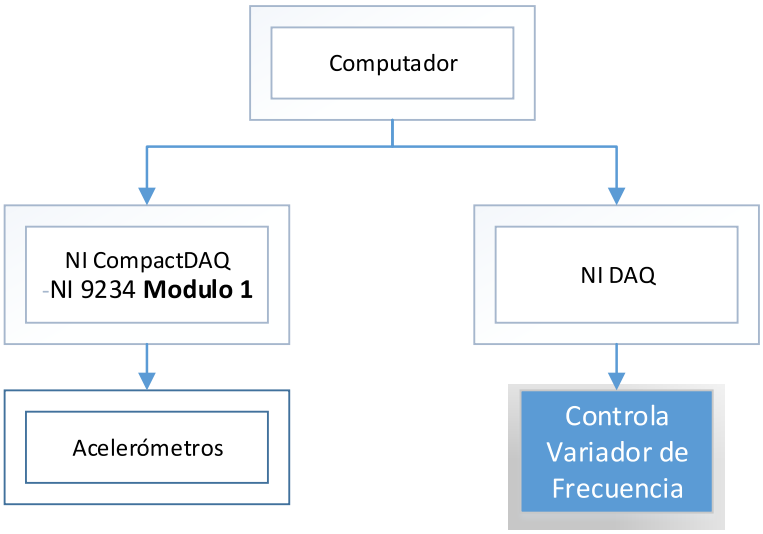


Figura 2.18: Sistema electrónico.

nuestro caso, se transmiten los datos hacia el variador de frecuencia). El software para controlar esta tarjeta tiene compatibilidad con los sistemas operativos Windows y algunas de las distribuciones Linux.

- **NI compactDAQ 9188**: Es un chasis para la adquisición de datos que tiene un nivel más robusto que los tradicionales, por lo que permite su uso en sistemas industriales. Es físicamente manejable, además posee conectividad y acondicionamiento de señales en entradas y salidas compuestas por diferentes módulos, que son usados para conectar directamente cualquier sensor.

  Un software desarrollado en LabView™, de National Instrument®(NI), se utiliza para modificar la manera en que se adquieren, analizan y presentan las señales, así como la forma en que se administran los datos de las medidas.

  Consta de 8 ranuras para conectar los dispositivos modulares, que pueden ser de más de 50 tipos diferentes dependiendo de las prestaciones que se necesiten. La comunicación entre el chasis y el computador se realiza mediante un cable ethernet, que ofrece mejores prestaciones que el cable USB en cuanto a robustez en los protocolos de comunicaciones y distancia permitida.

  Como elemento principal acoplado a este chasis está el módulo NI 9234, un módulo de adquisición de datos dinámicos que consta de 4 canales con los que se realizan mediciones de alta precisión. Cuenta con un rango dinámico de medición de 102dB, e incorpora un módulo de acondicionamiento para señal piezoeléctrica (IEPE).

  Estos 4 canales de entrada para las mediciones adquieren datos simultáneamente a tasas de velocidad de hasta $51{,}2kS/s$. Algunas especificaciones de este módulo son:

  - Marca: NI

- Modelo: NI9234
- Nº Canales: 4, 24 bits
- Frecuencia muestreo: 51.2kS/s
- Voltaje entrada: +-5V

- **Sensores de vibración o acelerómetros**: Los *acelerómetros*, también llamados *sensores de aceleración*, son instrumentos que realizan una medida de aceleración o vibración generando una señal eléctrica que mide una variación física como la vibración. Existen diferentes modelos, producto de combinar las diferentes tecnologías disponibles, como son los acelerómetros piezoeléctricos, los piezoresistivos, o los capacitivos.

  El más usado es el piezoeléctrico por su precisión en la medición, que usa un efecto descubierto en 1880 por Jacques y Pierre Curie. Este fenómeno se encontró en los cristales de cuarzo, donde se vio que al deformarlos por la acción de una fuerza se produce en ellos una polarización eléctrica que puede ser medida. Además, se descubrió que este efecto se puede realizar de manera reversible, es decir, que si se le aplica una carga eléctrica entre dos caras del material, éste se deforma.

  El cristal de cuarzo no es el único material piezoeléctrico. Se han encontrado estas características en materiales como la turmalina o la sal de Rochelle. Debido a que estos materiales, a pesar de su buena estabilidad ante el cambio y condiciones medioambientales, no producen una señal muy fuerte, su medición se hace difícil, y por ello se reemplazan por materiales sintéticos que, gracias a un tratamiento adecuado, son capaces de generar un potencial más fácil de medir bajo condiciones de deformación.

  El sensor utilizado para capturar las señales de vibración es un acelerómetro modelo 603C01 que tiene las siguientes características:

  - Marca: IMI SENSORS
  - Modelo: 603C01
  - Sensibilidad: 100mV/g o 10.2mV/(m/s2)
  - Rango medición: +-50g o +-490m/s
  - Frecuencia: 3 a 600.000 cpm (ciclos por minuto) o 0.5 a 10.000Hz

  Para la conexión de este sensor se emplea un cable BNC apantallado de 10ft.

2.5.2. Software de adquisición

En la Figura 2.19 se muestra el diagrama del proceso llevado a cabo en el software, cuyos detalles se presentan a continuación:

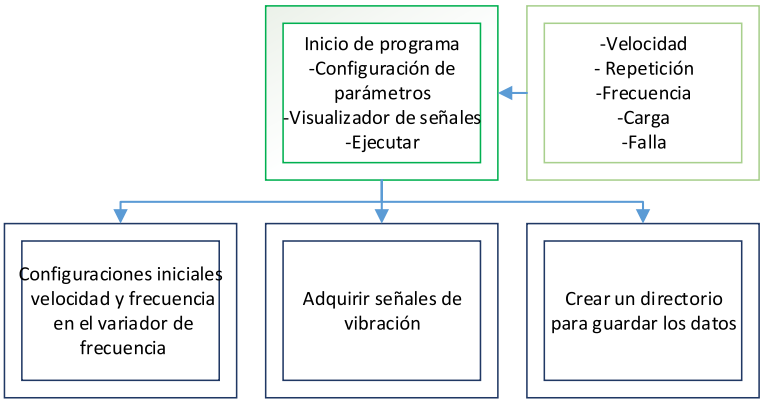


Figura 2.19: Software de adquisición.

Configuración de parámetros

La configuración de los parámetros consiste en establecer cuáles serán los valores de configuración que se enviarán al variador de frecuencia y a la fuente del freno magnético por medio de la tarjeta NI DAQ 6212, dependiendo de las pruebas que se vayan a realizar.

Para ello, los parámetros a establecer son:

- **Velocidad del motor**: La velocidad de un motor se mide como velocidad angular, es decir, el número de vueltas que da por unidad de tiempo, y se mide habitualmente en revoluciones por minuto (rpm) o radianes por segundo (rad/s). La velocidad del motor viene dada por la ecuación:

$$n = \frac{60 \cdot f}{p} \tag{2.14}$$

  Donde:

  - f es la frecuencia de la red,
  - n es el valor de la velocidad de sincronismo del motor, y
  - p es el número de polos del motor.

  El valor f puede ser modificado usando el variador de frecuencia y, como el número de polos es un valor fijo, la velocidad del motor es esencialmente proporcional a este valor de la frecuencia. Por ello, se usa la notación F para codificar la velocidad del motor.

- **Carga**: La carga es una resistencia al movimiento de rotación que se le da al sistema con la finalidad de simular el movimiento de alguna máquina o elemento que pueda estar conectada a la caja de engranes. Esta resistencia se logra conectando el sistema a un freno magnético regulable a través de un sistema de poleas. Notaremos por Ln cada nivel de carga en nuestros experimentos.

- **Repetición**: La repetición establece cuántos paquetes de datos con los mismos parámetros se capturarán. Por ejemplo, si se quieren tomar 5 muestras de 10 segundos con los mismos datos de velocidad y carga, las primeras señales adquiridas en los 10 segundos tendrán la etiqueta $R1$, las siguientes la etiqueta $R2$, y así sucesivamente.
- **Falla**: Para cada configuración experimental las fallas están detalladas en la Sección 2.6. Se denotará por P_n para indicar el n-ésimo tipo de falla considerado.

Configuraciones iniciales

Las configuraciones iniciales se realizan en el momento que se hace una primera comunicación entre el software de adquisición y el sistema electrónico. La ejecución del software de adquisición consta de dos pasos principales: el primer paso es la inicialización del programa en general, el segundo paso es la inicialización del programa de adquisición. Al realizar el encendido general se cargan los controladores respectivos y se envían los datos de configuración al variador de frecuencia, preparándolo para dar inicio al programa de adquisición y movimiento del sistema mecánico. La Figura 2.20 muestra un ejemplo del proceso de carga del parámetro de velocidad para el motor.

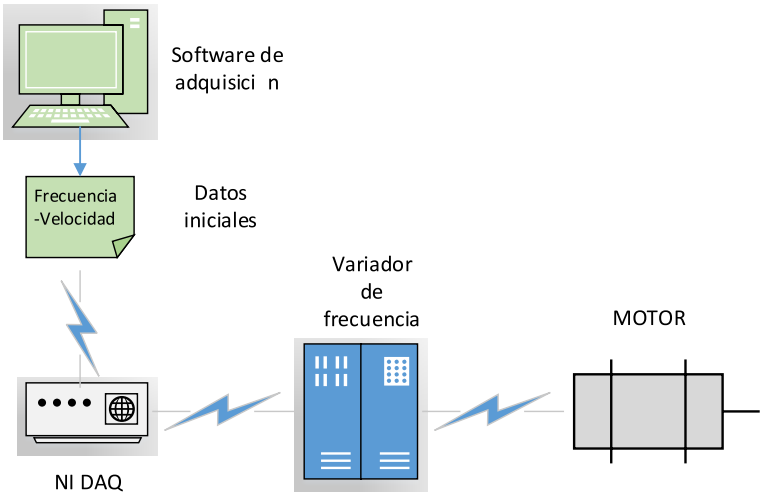


Figura 2.20: Proceso de carga de parámetros.

Adquisición de señales de vibraciones

Con la ejecución del programa comienza el proceso de adquisición de datos, para lo cual los sensores de aceleración conectados a la caja de engranes envían señales a las tarjetas de medición que se encuentran en cada uno de los módulos de la NI

compactDAQ, que como indicamos se comunica con el computador a través de una red ethernet. La adquisición se configura a una tasa de muestreo de $50kS/s$.

Debido a que los sensores utilizados tienen una sensibilidad específica, en cada sensor es necesario convertir los datos capturados para obtener la señal verdadera que el sensor está midiendo (ver Figura 2.21).

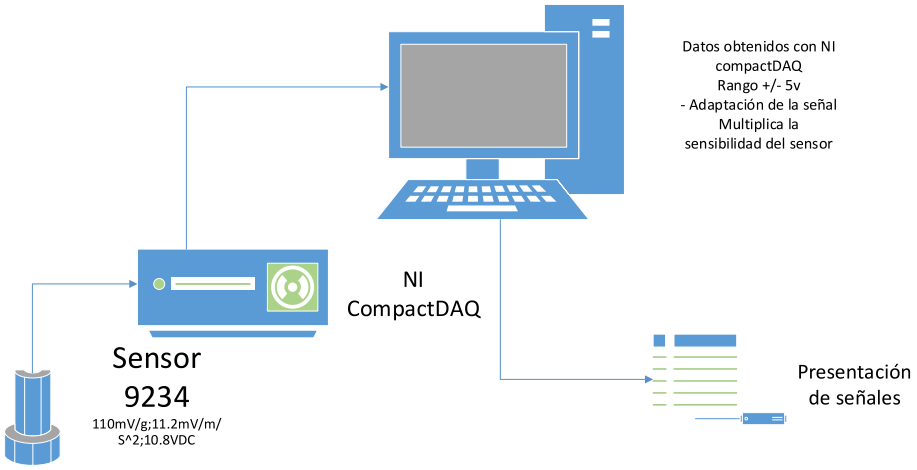


Figura 2.21: Proceso de adaptación de la señal.

La sensibilidad de un sensor viene dada por la razón entre el incremento de la lectura y el incremento de la variable que la ocasiona después de haberse alcanzado el estado de reposo. Este valor, que viene detallado en el sensor, debe ser multiplicado por el valor que captura el sensor para saber cuál es la medición real del fenómeno.

Creación de directorios y almacenamiento

Con el fin de guardar de forma ordenada los datos que se obtienen con las mediciones, se sigue un protocolo relativamente estándar: se ingresa una ruta principal donde se desea almacenar los datos, se crea una carpeta con un nombre específico (que normalmente detalla las características de la prueba que se está realizando, como velocidad, repetición, carga y falla), y dentro de esta carpeta se crea un archivo donde se guardarán lo datos obtenidos por cada sensor.

El software de adquisición genera un archivo .TDMS al recolectar los datos, que posteriormente debe ser convertido a .mat (formato propietario de MATLAB), para que sea fácilmente manipulable.

2.6. Configuración experimental para tareas CBM

Como hemos comentado, este trabajo muestra la aplicación de distintas propuestas de modelado de sistemas dinámicos a tareas específicas de detección y diagnóstico de fallos en elementos mecánicos. Con este fin se ha desarrollado una base experimental para la captura de datos desde un sistema mecánico rotativo diseñado e implementado en la Universidad Politécnica Salesiana de Cuenca (Ecuador) que permite utilizar elementos que se pueden ubicar siguiendo distintas configuraciones con el fin de simular mecanismos encontrados en la industria bajo condiciones de funcionamiento (y que pueden ser constantes o variables a lo largo del tiempo).

2.6.1. Configuración del banco de vibraciones para diagnóstico de fallos en cajas de engranes rectos

La configuración del banco de vibraciones para el diagnostico de fallos en engranes rectos cuenta con un motor trifásico de corriente alterna que se encuentra acoplado a una caja reductora de engranes rectos con una sola etapa, y en su salida posee una polea para acoplar un freno magnético que simula una carga (ver Figura 2.22).

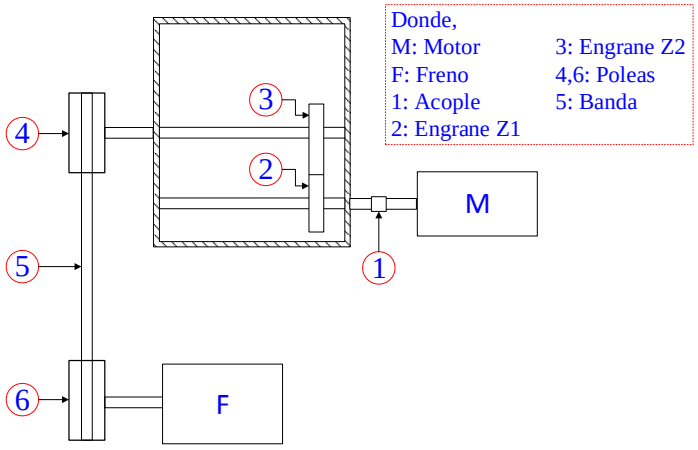
Para la simulación de fallos se cuenta con engranes rectos, en los cuales se generaron fallos por fisuras, dientes rotos, picados y desgaste, abarcando así la mayor parte de fallos que se presentan en estos elementos (ver Tabla 2.1).

2.6.2. Configuración del banco de vibraciones para severidad en engranes helicoidales

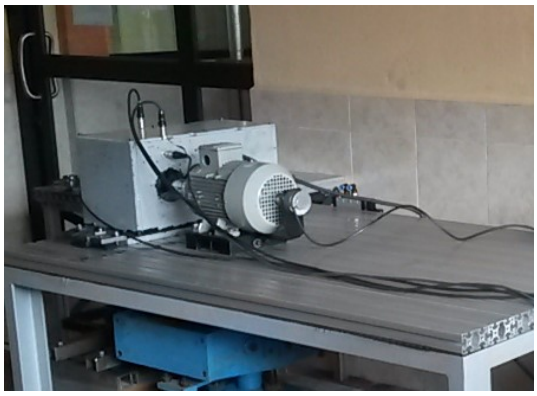
Para el análisis de severidad se ha utilizado un motor trifásico de corriente alterna acoplado a una caja reductora de una etapa y transmisión al freno magnético por medio de bandas, como se muestra en la Figura 2.23.

La simulación de daños en engranes helicoidales se realizó con fallos por diente roto, que es el fallo más común en cajas reductoras y que normalmente se presenta en el engrane de menor número de dientes ($Z1$) puesto que se somete a un mayor esfuerzo (hasta el punto de poder presentar un diente roto al 100 %). En la Tabla 2.2 se muestran las características de cada nivel de severidad, como son la masa del engrane y la longitud del diente.

En la Figura 2.24 se pueden observar engranes helicoidales que presentan fallos que van desde un 10 % hasta el 100 % (codificados con P2 hasta P10, respectivamente).



(a) Disposición de elementos



(b) Montaje de elementos

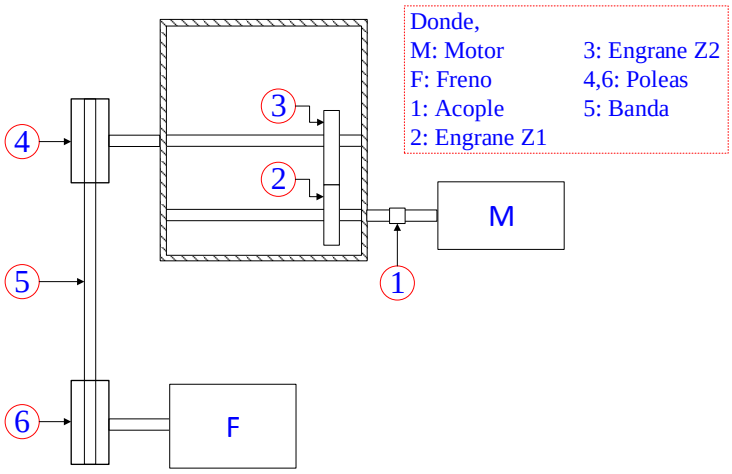
Figura 2.22: Configuración de banco de vibraciones para el diagnóstico de fallas en engranajes rectos.

Fallo	Característica	Código
Sin fallo	-	P1
Grieta	Toda la raíz de un diente Ancho 0.7mm Profundidad 0.4mm	P2
Diente roto	Diente roto 10 %	P3
Picaduras	Picaduras en todos los dientes Tres picaduras por diente Diámetro 1mm Profundidad 0.5 mm	P4
Desgaste	Línea en cara de diente Profundidad 0.2mm	P5
Desalineamiento	Entre ejes de 1°	P6
Desalineamiento	Entre ejes de 2°	P7
Desalineamiento	Entre ejes de 3°	P8
Diente roto	Diente roto 50 %	P9
Diente roto	Diente roto 100 %	P10

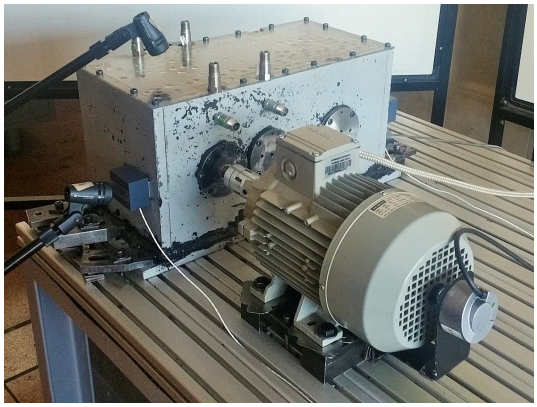
Cuadro 2.1: Características de fallos en engranes rectos.

Código	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Porcentaje de rotura [%]	0	10	20	30	40	50	60	70	85	100
Masa [g]	575.7	562.4	553.4	543.4	532.9	516.2	506.0	494.9	477.1	460.6
Longitud de diente [mm]	20.4	18.3	16.3	14.2	12.2	10.2	8.1	6.1	4	0

Cuadro 2.2: Características de severidad en engranes helicoidales.



(a) Disposición de elementos



(b) Montaje de elementos

Figura 2.23: Configuración del banco de vibraciones para el estudio de la severidad de fallo en engranes helicoidales.

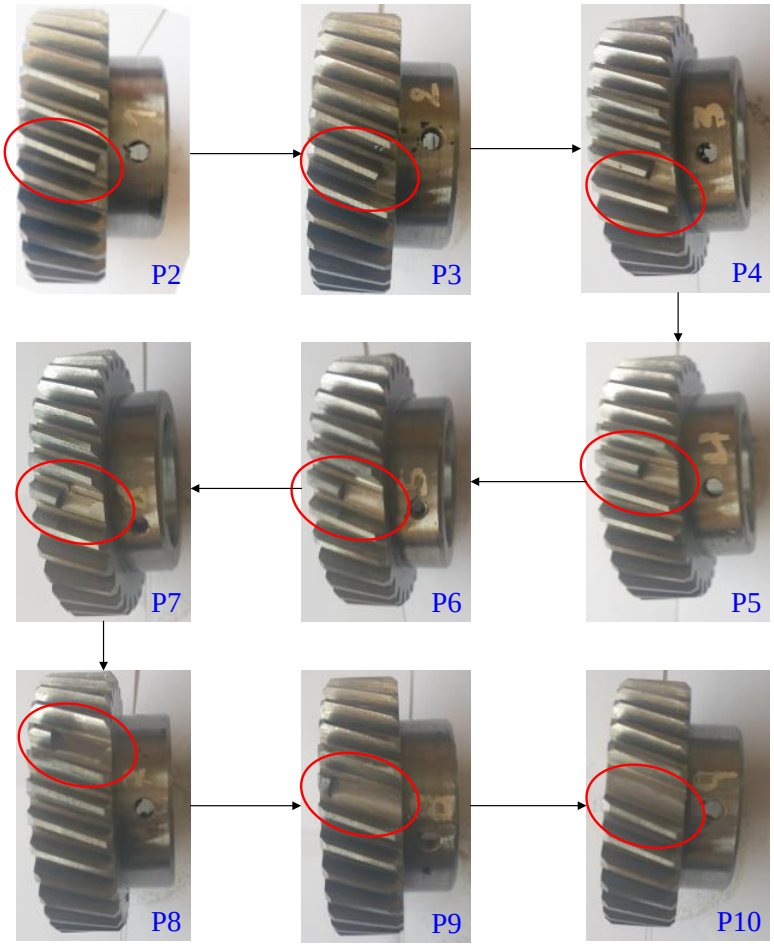
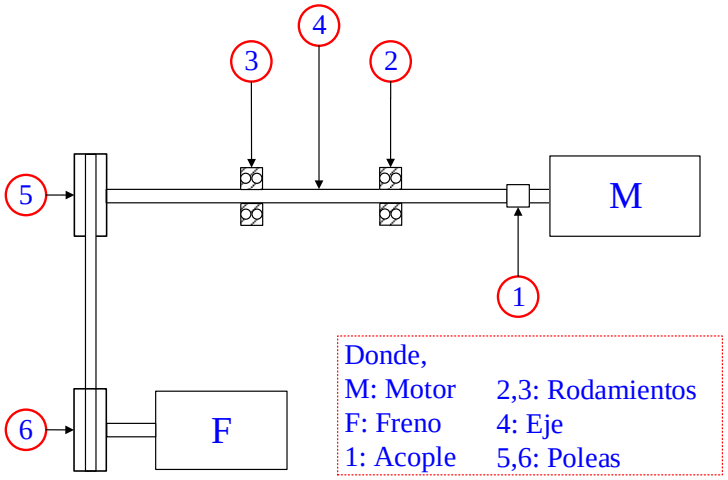


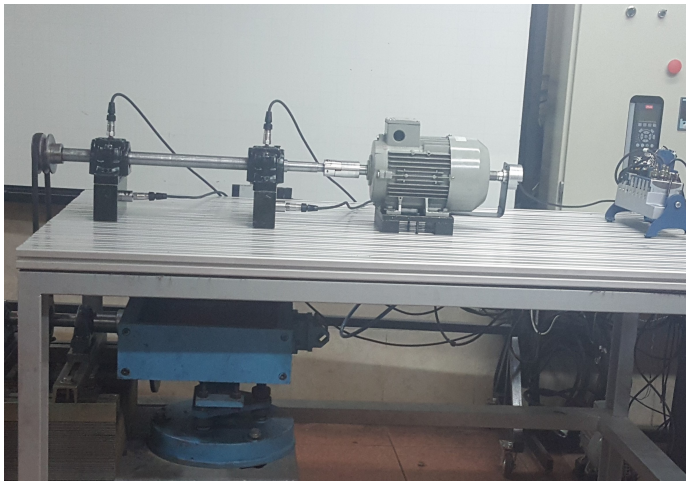
Figura 2.24: Severidad en engranes helicoidales.

2.6.3. Configuración del banco de vibraciones para severidad en rodamientos

Para la simulación de severidad de fallos en rodamientos se ha utilizado un motor trifásico de corriente alterna como elemento motriz, que transmite su potencia a un eje que cuenta con dos rodamientos, donde en uno de ellos se evalúa la severidad del daño en sus distintos componentes (ver Figura 2.25). A continuación se presentan los detalles de las modificaciones realizadas para simular los fallos en cada elemento del rodamiento.



(a) Disposición de elementos



(b) Montaje de elementos

Figura 2.25: Configuración del banco de vibraciones para severidad en rodamientos.

Pista interna

Para la simulación de fallos en la pista interna se consideran los tres niveles de severidad que se presentan en la Tabla 2.3.

Código	Características de fallo	
	Diámetro [mm]	Profundidad [mm]
P2	0.5	0.3
P3	0.9	0.3
P4	1.3	0.3

Cuadro 2.3: Características de severidad en pista interna de rodamientos.

Cada uno de estos fallos se ha construido por medio de electro-erosión, que consiste en usar corriente eléctrica para cortar a precisión el material. La Figura 2.26 muestra los distintos niveles de severidad considerados.

Figura 2.26: Niveles de severidad en pista interna de rodamientos.

Pista externa

Para el estudio de la severidad de fallo en la pista externa se han considerado los tres niveles de daño que se detallan en la Tabla 2.4, y que simulan un desgaste en la pista externa a causa de un elemento rodante defectuoso. En la Figura 2.27 se muestran imágenes con estos fallos.

Código	Características de fallo	
	Diámetro [mm]	Profundidad [mm]
P5	0.5	0.3
P6	0.9	0.3
P7	1.3	0.3

Cuadro 2.4: Características de severidad en pista externa de rodamientos.

Figura 2.27: Niveles de severidad en pista externa de rodamientos.

Elemento rodante

Para el elemento rodante se han considerado 10 niveles de severidad, tal y como detalla la Tabla 2.5. La Figura 2.28 muestra el resultado del proceso de construcción para cada uno de los niveles de severidad.

Código	Características de fallo		
	Diámetro [mm]	Profundidad [mm]	Masa [g]
P8	1	0.5	2.70103
P9	0.9	0.5	2.70250
P10	0.8	0.5	2.70300
P11	1	0.3	2.70327
P12	0.9	0.3	2.70347
P13	0.8	0.3	2.70353
P14	0.7	0.3	2.70417
P15	0.6	0.3	2.70460
P16	0.6	0.1	2.70460
P17	0.5	0.1	2.70473

Cuadro 2.5: Características de severidad en elemento rodante.

2.6.4. Configuración del banco de vibraciones para diagnóstico multi-falla en rodamientos

Para la tarea de diagnóstico multifalla en rodamientos se ha utilizado un motor trifásico de corriente alterna que transmite su movimiento rotacional a un eje por medio de un acoplamiento directo. A lo largo del eje se encuentran dos rodamientos en los que se configuran distintos tipos de daños en sus elementos, los cuales pueden presentarse de forma individual (en cada rodamiento) o de forma conjunta (dos rodamientos a la vez). La Tabla 2.6 muestra las diferentes configuraciones evaluadas.

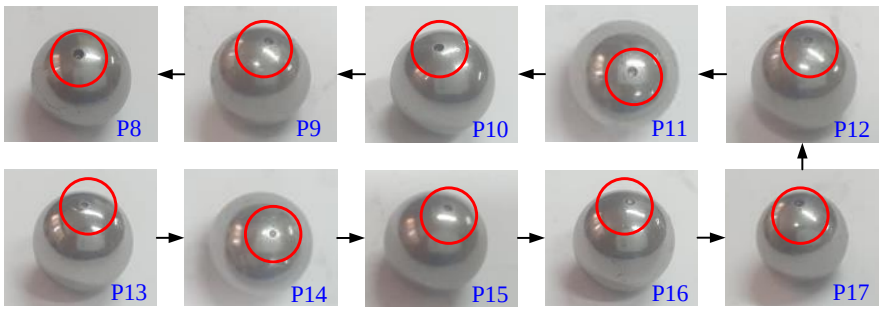


Figura 2.28: Niveles de severidad en elemento rodante de rodamientos.

Código	Elementos evaluados	
	Rodamiento 1	**Rodamiento 2**
P1	normal	normal
P2	fallo en pista interna	normal
P3	fallo en pista externa	normal
P4	fallo en elemento rodante	normal
P5	fallo en pista interna	fallo en pista externa
P6	fallo en pista interna	fallo en elemento rodante
P7	fallo en elemento rodante	fallo en elemento rodante

Cuadro 2.6: Características de experimentación multi-fallas en rodamientos.

Adicionalmente, se simulan múltiples condiciones de funcionamiento de carga y velocidad. Para generar distintas condiciones de carga se utilizan dos ruedas volantes que pueden colocarse en el espacio entre los dos rodamientos. De esta manera se simulan 3 condiciones de carga: sin rueda, 1 rueda y 2 ruedas. En el mismo sentido, se generan 3 velocidades de rotación para el motor: 8Hz, 10Hz y 15Hz. La configuración mecánica descrita anteriormente puede ser vista en la Figura 2.25.

2.7. Aprendizaje Automático (Machine Learning)

El *Aprendizaje Automático* (Machine Learning, por su nombre en inglés, que está mucho más extendido actualmente[3]) es una rama de la Inteligencia Artificial, de difícil definición (como la propia Inteligencia Artificial) y que, grosso modo, tiene como objetivo desarrollar técnicas que permitan a las computadoras aprender.

[3] A lo largo de este trabajo usaremos indistintamente la denominación española y la inglesa.

Para conseguir este objetivo, una de las tareas principales en el aprendizaje automático es crear algoritmos que sean capaces de generalizar comportamientos y reconocer patrones a partir de información proporcionada por medio de ejemplos. En este sentido, este área proporciona un conjunto de métodos para el análisis de datos, lo que hace que se solape con otras áreas provenientes de la estadística, como son la minería de datos y el aprendizaje estadístico. La diferencia natural entre estas últimas aproximaciones y la del aprendizaje automático es el componente central del algoritmo como herramienta computacional, y no como un mero traductor de métodos de aproximación numérica. Es por ello que, en este sentido, el aprendizaje automático suele ofrecer mayor potencia y flexibilidad que los anteriores, pues no está restringido a las teorías estadísticas en las que se suelen basar las demás. El aprendizaje estadístico crea sus modelos en base a teorías previas y suposiciones sobre la distribución de probabilidad del conjunto de datos, mientras que los modelos de aprendizaje automático no siempre pueden ser justificados de forma teórica ni asumen distribuciones funcionales sobre los datos. El análisis estadístico prueba sus modelos por medio de tests de hipótesis mientras que el aprendizaje automático lo hace con el uso de métricas de desempeño sobre instancias conocidas del problema.

En una primera aproximación, podríamos decir que una de las tareas del aprendizaje automático es intentar extraer conocimiento sobre algunas propiedades no observadas de un objeto (que viene dado en forma de dato) basándose en las propiedades que sí han sido observadas de ese mismo objeto (o incluso en propiedades observadas en otros objetos similares). En otras palabras, convierte el proceso de aprender en el de predecir comportamiento futuro (desconocido) a partir de lo que ha ocurrido en el pasado (conocido). Por ello, muchas de las soluciones aportadas por el aprendizaje automático tienen como finalidad automatizar el proceso de construcción de modelos analíticos y algorítmicos a partir únicamente de la disposición de un conjunto de datos. El modelo obtenido en el proceso (que se conoce como *aprendizaje*) debe lograr la generalización del patrón observado en los datos.

Aunque el aprendizaje automático no es una campo de estudio nuevo, es en los últimos años cuando, gracias a la potencia computacional de las nuevas tecnologías en ordenadores, ha resurgido proporcionando innumerables aplicaciones y con un ritmo de generación y novedad pocas veces observado en otras áreas de investigación. Por ejemplo, al tiempo de la escritura de esta memoria, ha sido posible diseñar coches de conducción automática que tienen como núcleo principal algoritmos de aprendizaje, se han implementado sistemas de reconocimiento de audio y traducción automática en tiempo real, se ha aplicado al análisis de datos astronómicos masivos para el descubrimiento de nuevas estrellas, ha aparecido el primer jugador de Go de nivel profesional (superando a los maestros del juego como en su día hiciera Deep Blue con el ajedrez), y cada día aparecen aplicaciones nuevas, en áreas completamente diversas, que van desde el ocio hasta aplicaciones médicas, pasando por control de robots y predicción de sistemas complejos.

A continuación vamos a presentar algunos conceptos habituales del Aprendizaje Automático, entre los que veremos especificaciones acerca del conjunto de datos y sus componentes, los tipos de aprendizaje que podemos encontrar, y la evaluación

Instancia	C_1	C_2	...	C_j	...	C_n
D_1	V_1^1	V_1^2	...	V_1^j	...	V_1^n
D_2	V_2^1	V_2^2	...	V_2^j	...	V_2^n
...	...	...	...	...	...	...
D_i	V_2^1	V_2^2	...	V_i^j	...	V_i^n
...	...	...	...	...	...	...
D_m	V_m^1	V_m^2	...	V_m^j	...	V_m^n

Cuadro 2.7: Representación de un conjunto de datos **D**.

del desempeño en los algoritmos aplicados sobre problemas concretos.

2.7.1. Conjunto de datos

El *conjunto de datos* (*dataset*, por su nombre en inglés[4]) es un grupo de instancias o ejemplos de un problema equivalente a una muestra en el análisis estadístico. Habitualmente, cada ejemplo del conjunto de datos está a su vez compuesto por un grupo de características específicas y comunes para todas las instancias. Las características cumplen con la función de distinguir a cada instancia como similar o diferente al resto, y es a partir de estas características que se buscan relaciones entre los elementos del conjunto de datos.

La forma más común de representar un conjunto de datos es en forma tabular, donde cada fila, i, representa una instancia, D_i, y cada columna, j, representa una característica, C_j. Se puede ver un ejemplo en la Tabla 2.7, donde se presenta un conjunto de datos **D** con m instancias, cada una de ellas haciendo uso de las mismas n características. En este sentido, cada característica es equivalente a una variable aleatoria en el análisis estadístico. Los valores que estas características pueden tomar para cada una de las instancias pueden pertenecer a un conjunto finito, en cuyo caso decimos que la variable asociada es *categórica*, o infinito, pudiendo ser *continua* o no, de manera similar a las variables categóricas, discretas y continuas del análisis estadístico. En lo que sigue, los términos *variable* y *característica* serán utilizados de forma intercambiable.

Concretamente, una característica categórica es aquella que puede tomar elementos de un conjunto finito (por ejemplo, el conjunto $Colores = \{amarillo, azul, rojo\}$), o infinito contable (por ejemplo, el conjunto de los números naturales $\mathbb{N}$). Las varia-

[4]Debido a que la mayor parte de la literatura sobre Aprendizaje Automático está únicamente en inglés, es común no tener traducciones estandarizadas de los términos utilizados en el área, por ello, y con el fin de facilitar la lectura de esta tesis a los que se acerquen a ella desde dentro del área, cometeremos muchas veces la ligereza de utilizar la denominación inglesa que comúnmente se encuentra en la literatura disponible.

bles categóricas se dividen en dos grupos:

- Categóricas Ordinales: que guardan un criterio de orden para sus elementos (por ejemplo, el conjunto $Medida = \{bajo, medio, alto\}$).

- Categóricas Nominales: que no guardan ninguna relación de orden entre sus elementos (por ejemplo, el conjunto $Alimentacion = \{herbivora, omnivora, carnivora\}$).

2.7.2. Tipos de Aprendizaje

Tal y como se mencionó anteriormente, las aplicaciones donde el Aprendizaje Automático está presente son muy diversas, así como la forma de entender qué tareas se pueden resolver por medio del aprendizaje. Una forma de encontrar la técnica correcta para una aplicación es determinar el tipo de aprendizaje requerido acorde a la información que se tiene en el conjunto de datos y al objetivo que se persigue.

Aunque la clasificación de los tipos de aprendizaje que damos a continuación no es la única posible, sí que podemos decir que es la más aceptada dentro del área:

- **Aprendizaje Supervisado**: se genera una función que establece una correspondencia entre las entradas y las salidas deseadas del sistema, donde la base de conocimientos está formada por ejemplos etiquetados a-priori (es decir, ejemplos de los que sabemos su salida correcta).

- **Aprendizaje No-Supervisado**: el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formados únicamente por entradas al sistema, sin conocer una salida correcta, por lo que se busca que el modelo sea capaz de reconocer patrones para poder dar criterios de semejanza entre los datos conocidos.

- **Aprendizaje por Refuerzo**: el algoritmo aprende observando el mundo que le rodea y con un continuo flujo de información en las dos direcciones (del mundo a la máquina y de la máquina al mundo) realizando un proceso de ensayo-error, y reforzando aquellas acciones que reciben una respuesta positiva en el mundo.

En los siguientes parágrafos concretaremos lo que se entiende por aprendizaje supervisado y no-supervisado, ya que serán utilizados como parte fundamental de este trabajo, pero debido a que no hacemos uso del aprendizaje por refuerzo no profundizaremos más en él, y queda a criterio del lector ahondar en esta interesante vía de generar algoritmos de aprendizaje.

Aprendizaje supervisado

El aprendizaje supervisado es una tarea que se realiza en aplicaciones donde el objetivo principal es aprender una función desconocida que toma como parámetros de entrada un grupo de características del conjunto de datos, comúnmente llamadas *variables de entrada o predictoras*, y considera como salida otra característica del mismo conjunto de datos, llamada *variable de salida u objetivo*.

Formalizando esta idea, podríamos escribir:

$$\hat{y} = f(\mathbf{x}),\ \forall(\mathbf{x}, y) \in \mathbf{D} \tag{2.15}$$

donde $\mathbf{x}$ es el vector de variables de entrada, f es la función que conseguimos aprender y, por tanto, $\hat{y}$ es la predicción que devuelve f y que debe ser lo más cercana posible al valor verdadero, y, para todas las instancias $\mathbf{x}$ del conjunto de datos, $\mathbf{D}$.

Como se puede apreciar en la ecuación (2.15), la característica fundamental del aprendizaje supervisado es que para cada instancia del conjunto de datos se tiene la tupla de la forma $(\mathbf{x}, y)$, que indica la salida deseada para cada dato de entrada.

Dependiendo del tipo de la variable de salida, el aprendizaje supervisado se suele clasificar a su vez en dos tipos:

- Clasificación: La variable de salida es categórica.
- Regresión: La variable de salida es continua.

En la tarea de clasificación, cada valor que puede tomar la variable de salida se denomina *clase*. Hablaremos de *Clasificador Binario* cuando el número de clases posibles es únicamente dos, y *Clasificador Multi-clase* cuando tenemos más de dos clases. Obsérvese que no tiene sentido aprender de problemas que tienen una sola clase, ya que se convierte en un problema trivial, aunque el último capítulo de esta tesis profundiza en el problema de aprendizaje de una clase, one-class learning, pero entendiéndolo como que hay varias clases posibles para aprender, pero a la máquina solo se le muestran ejemplos pertenecientes a una de ellas, por lo que pasa de ser un problema de clasificación supervisado a uno no-supervisado.

Volviendo al área de aplicación que nos interesa en esta memoria, un ejemplo de tarea que requiere de un aprendizaje supervisado es el diagnóstico automático de fallas de un sistema. Para la resolución de esta tarea se requiere de un conjunto de datos donde cada instancia esté compuesta por un vector de entrada $\mathbf{x}$, con un síntoma por cada elemento del vector, y un valor y con la posible falla asociada al conjunto de mediciones $\mathbf{x}$. El objetivo es encontrar una función f que establezca la relación entre el conjunto de mediciones y la falla mediante el uso de algoritmos de aprendizaje sobre el conjunto de datos, que intentan descubrir la relación existente entre los predictores y el objetivo.

El modelo final que representa la función f aprendida puede ser entonces utilizada con nuevas instancias del problema, ya sea para realizar pruebas adicionales y

comprobar la robustez del modelo, o para realizar predicciones reales de una falla en base a un conjunto de mediciones y actuar como asistente de diagnóstico.

Aprendizaje no-supervisado

El aprendizaje no-supervisado, al igual que el supervisado, busca aprender una función a partir de los datos disponibles. Sin embargo, como hemos comentado anteriormente, se diferencian en el hecho de que para el no-supervisado no se cuenta con la tupla de pares (entrada-salida), $(\mathbf{x}, y)$, para los ejemplos, por lo que el grupo de aplicaciones que pueden ser abordadas con este tipo de aprendizaje es distinto.

Entre los principales problemas abordados con el aprendizaje no-supervisado está el de la formación de grupos de instancias relacionadas, comúnmente llamado *clustering de datos*. Las relaciones que buscan los algoritmos para clustering utilizan la información de las características, que muchas veces suele ser información de posición proveniente de una representación de las instancias en un espacio geométrico. El problema, entonces, se traduce en conseguir una inmersión adecuada de los datos de aprendizaje en un espacio geométrico con una métrica adecuada que refleje las relaciones de similitud entre los datos como indica el problema.

Un problema relacionado que se intenta resolver con este tipo de aprendizaje es el de *detección de anomalías*. El objetivo es aprender un modelo que sea capaz de extraer la relación existente entre las instancias de un conjunto de datos (que se supone que forman un solo grupo). De esta forma, posteriormente, cualquier instancia nueva que no cumpla con la relaciones encontradas por el modelo será clasificada como una instancia anómala.

2.7.3. Evaluación de desempeño

Como hemos visto, en general las técnicas de aprendizaje automático construyen modelos a partir de un conjunto de datos. Como un ejemplo concreto para el aprendizaje supervisado hemos visto que estos modelos proporcionan funciones de predicción. Sin embargo, al no tener un sustento teórico basado en leyes físicas asociadas al proceso que generó los datos ni al algoritmo aplicado para la generación del modelo, la validez de los resultados no podrá ser probada de forma analítica.

Antes de mostrar las opciones disponibles para resolver el problema de la validación de un modelo, daremos algunas definiciones importantes:

- Modelo congruente: Es un modelo que predice de manera correcta todas las instancias utilizadas para su generación.
- Modelo con capacidad de generalización: Es un modelo que predice de manera correcta instancias nuevas que no aparecen en el conjunto de datos utilizado para su generación.

- Modelo sobre-ajustado: Es un modelo que tiene la capacidad de predecir de forma correcta los ejemplos del conjunto con el que fue entrenado, pero no es capaz de predecir de forma correcta instancias nuevas (es congruente, pero no generaliza bien).

El ideal, por supuesto, es un modelo que tenga capacidad de generalización perfecta, es decir, que a partir de un conjunto (más o menos reducido) de ejemplos de entrada, sea capaz de predecir correctamente el resto de posibles ejemplos del problema. Debido a que este ideal es en la mayoría de los casos (al menos en los interesantes) inalcanzable, hemos de asumir como cierto (aunque no necesariamente lo es) que si el modelo es capaz de predecir un conjunto suficientemente grande de instancias nuevas entonces tiene capacidad de generalización y por lo tanto es correcto (o, al menos, útil). Esta afirmación se debe a que un conjunto lo suficientemente grande de instancias nuevas debería contener una diversidad representativa del espacio completo de características.

Se ha mencionado que la afirmación anterior no necesariamente es cierta porque, aunque se disponga de un número muy grande de instancias nuevas, no se puede asegurar nada sobre su diversidad y en la mayoría de aplicaciones, sobre todo aquellas donde las variables son continuas, no se puede disponer del dominio completo de dichas variables. La realidad suele ser que, tanto para la construcción del modelo como para la validación del mismo, se cuente únicamente con un conjunto de datos para las dos tareas.

Partición binaria del conjunto de datos y holdout

En vista de la necesidad de utilizar un solo conjunto de datos para generar y validar un modelo de aprendizaje, se deben plantear estrategias para dividirlo de tal forma que esto influya lo menos posible en la calidad del modelo resultante. El objetivo de esta división, además de conseguir dos subconjuntos, uno para entrenamiento y otro para verificación del modelo, es conseguir que estos tengan la mayor diversidad con respecto a las clases existentes y al espacio de representación de características.

El principal inconveniente que se puede presentar con la generación del modelo es sesgarlo (dar preferencia) hacia alguna clase si el conjunto de datos con el que se crea el modelo tiene mayor cantidad de instancias de dicha clase. Algo que, al mismo tiempo, perjudica a las clases con menor número de instancias en el conjunto de datos.

Una de las estrategias mas simples, comúnmente llamada *holdout*, consiste en realizar una división directa del conjunto de datos en dos subconjuntos, el de entrenamiento y el de prueba. El conjunto de entrenamiento será utilizado junto con el algoritmo de aprendizaje para generar el modelo a partir de los datos, y posteriormente el conjunto de prueba (que no se ha usado previamente en ninguna de las etapas de entrenamiento) servirá para evaluar el modelo generado.

Ambos conjuntos son generados a partir de un proceso de muestreo estratificado y aleatorio del conjunto de datos para mantener la mayor diversidad en los subconjuntos resultantes. La estratificación se realiza dividiendo inicialmente el conjunto de datos en tantos subconjuntos como clases existan (llamados *estratos*). Luego, de cada uno de estos subconjuntos se toma, de forma aleatoria y sin reemplazo, un porcentaje de instancias que pasará a formar parte del conjunto de entrenamiento. Las instancias restantes de los subconjuntos iniciales formarán parte del conjunto de prueba. De esta forma se garantiza la mayor diversidad (un número similar de instancias de cada clase) para el proceso de entrenamiento y el de prueba, evitando sesgar el modelo a favor de una clase específica.

Habitualmente, el conjunto de prueba se suele considerar más pequeño que el conjunto de entrenamiento, ya que la potencia de un modelo basado en datos está relacionada de forma directa con la cantidad de datos con que se ha entrenado. En la literatura se reporta un uso común de $70\,\%$ para entrenamiento y $30\,\%$ para prueba. Un ejemplo de holdout para un conjunto de datos con 4 clases se muestra en la Figura 2.29.

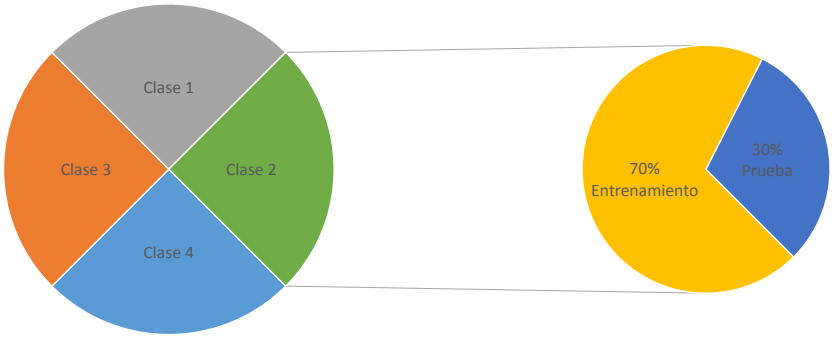


Figura 2.29: Ejemplo de holdout aplicado a un conjunto de datos con 4 clases para una partición de $70\,\%$ para entrenamiento y $30\,\%$ para prueba.

k-particiones del conjunto de datos y validación cruzada

El método de partición anterior resulta útil cuando el conjunto de datos es lo suficientemente grande como para que no importe la pérdida de los datos que se destinan al conjunto de prueba y cuando la varianza estimada del modelo no sea un elemento importante en la evaluación. En estas condiciones, *suficientemente grande* quiere decir que haya un número representativo de instancias de la clase con menor número de elementos y, además, que el número de clases sea pequeño en relación al número total de instancias. Cuando estas condiciones no se cumplen es conveniente usar otro método de evaluación que permita utilizar una cantidad de datos más pequeña para el proceso de evaluación y a su vez disminuya la varianza estimada del modelo.

Un procedimiento útil para este caso empieza realizando k particiones (k-folds) en el conjunto de datos. Estas particiones, al igual que en el caso anterior, se obtienen por un proceso de estratificación. Para cada partición de las obtenidas se entrena el modelo con las restantes $k-1$ particiones y se verifica con la partición sobrante. A este procedimiento se le conoce como *validación cruzada*.

De esta forma se obtiene una métrica de evaluación para cada partición del conjunto de datos, habiendo logrado entrenar y probar el modelo con todo el conjunto de datos. Tras este procedimiento es común obtener una métrica general de desempeño del modelo promediando los resultados de evaluación en cada partición. También es posible obtener la varianza estimada en el proceso de evaluación, que disminuirá a medida que el número de particiones sea incrementado.

A pesar de que el método de k-particiones con validación cruzada es más eficaz y da información más completa de la evaluación de un modelo, no siempre es aplicable, ya que el coste computacional del proceso de aprendizaje puede ser tan elevado que repetir k veces el algoritmo completo puede ser desaconsejable desde un punto de vista práctico. Es por ello que este método se utiliza cuando el conjunto de datos es relativamente pequeño, y normalmente con un valor de k no mayor que 10.

2.7.4. Métricas de evaluación

Hasta ahora se ha considerado la evaluación de un modelo con una métrica abstracta capaz de definir su desempeño. Sin embargo, desde un punto de vista práctico es importante conocer qué métricas se deben utilizar para cada tarea concreta. En esta sección vamos a definir las métricas más comunes de acuerdo al tipo de aprendizaje y la información que buscamos conocer sobre un modelo.

Métricas de clasificación

La mayoría de métricas de desempeño usadas en problemas donde se conoce la variable de salida, y ésta es categórica, pueden ser calculadas desde la información proporcionada por una tabla llamada *matriz de error* o *matriz de confusión*, que se construye a partir de los errores producidos en la predicción realizada por el modelo, $\hat{y}$, con respecto al valor real, y. En el caso categórico suele considerarse un error en una predicción cuando $\hat{y} \neq y$. Las filas de esta matriz representan las clases reales y las columnas representan las predicciones de las clases en el conjunto de prueba. La Tabla 2.8 muestra una matriz de confusión para un caso de tres clases donde, por ejemplo, el valor 6 de la fila 1 y la columna 2 indica que existen 6 elementos del conjunto de prueba que fueron clasificados como clase 2 por el modelo cuando realmente pertenecen a la clase 1.

De una matriz de confusión M de tamaño $n \times n$ que muestra los resultados de un problema de clasificación con n clases se puede obtener un grupo de valores que definen los errores y aciertos cometidos por un modelo en una clase concreta. Nos

Clase	Predicción Clase 1	Predicción Clase 2	Predicción Clase 3
Clase 1	575	6	3
Clase 2	6	572	1
Clase 3	2	3	557

Cuadro 2.8: Ejemplo de matriz de confusión para una evaluación del conjunto de prueba de 3 clases.

referimos al número de *verdaderos positivos*, *verdaderos negativos*, *falsos positivos* y *falsos negativos*, que se detallan a continuación:

- **Verdaderos positivos**(t_{pos}): es el número de instancias de la clase que fueron clasificadas de forma correcta en ella. Para una clase c concreta es el número que aparece en la posición (c, c) de la matriz de confusión:

$$t_{pos}(c) = M(c, c) \tag{2.16}$$

- **Verdaderos negativos**(t_{neg}): es el número de instancias que acertadamente no fueron clasificadas en la clase. Es la suma los elementos de la diagonal principal, excepto el elemento que se encuentra en la fila y columna c:

$$t_{neg}(c) = \sum_{i=1, i \neq c}^{n} M(i, i) \tag{2.17}$$

- **Falsos positivos**(f_{pos}): es el número de instancias que de forma errónea fueron clasificadas como pertenecientes a la clase. Para una clase c es la suma de los elementos de la columna c, excepto el de la fila c:

$$f_{pos} = \sum_{i=1, i \neq c}^{n} M(i, c) \tag{2.18}$$

- **Falsos negativos**(f_{neg}): es el número de instancias de la clase que fueron clasificadas de forma incorrecta en el resto de clases. Es la suma de los elementos de la fila c de la matriz de confusión a excepción del elemento de la columna c:

$$f_{pos} = \sum_{i=1, i \neq c}^{n} M(c, i) \tag{2.19}$$

Estos valores pueden ser obtenidos para cada una de las clases que definen al problema, y a partir de ellos es posible calcular un conjunto de métricas de desempeño del modelo para cada clase. Las principales son:

- **Recall (Sensibilidad):** indica la proporción de instancias que fueron correctamente clasificadas por el modelo para una clase concreta en relación al número total de elementos en el conjunto de prueba que realmente pertenecen a dicha clase:

$$Recall = \frac{t_{pos}}{t_{pos} + f_{neg}} \tag{2.20}$$

- **Precision (Precisión):** indica la proporción de instancias que fueron correctamente clasificadas por el modelo para una clase concreta en relación al número total de elementos en el conjunto de prueba que el modelo predijo como pertenecientes a esa clase:

$$Precision = \frac{t_{pos}}{t_{pos} + f_{pos}} \tag{2.21}$$

- **F-value (Valor F):** fusiona las dos métricas anteriores en un solo valor para tener en cuenta la proporción de aciertos comparados tanto con los falsos positivos como con los falsos negativos:

$$ValorF = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.22}$$

- **Accuracy (Exactitud):** indica la proporción de instancias que fueron correctamente clasificadas por el modelo en relación al número total de elementos en el conjunto de prueba:

$$Accuracy = \frac{t_{pos} + t_{neg}}{t_{pos} + t_{neg} + f_{pos} + f_{neg}} \tag{2.23}$$

  Como se aprecia en la ecuación, esta es una métrica que tiene el mismo resultado para todas las clases y por lo tanto permite tener una visión general del desempeño global del modelo.

Métricas para regresión

En las tareas en donde se requiere que la variable de salida sea continua o discreta ordinal y la distancia de la predicción con el valor real representa un grado de error, las métricas anteriores no muestran adecuadamente la viabilidad del modelo. Para estos casos es necesario definir otras métricas que contemplen no solo si el valor de salida es o no igual al valor real sino la distancia[5] entre los dos.

A continuación vamos a definir las métricas más usadas para la tarea de regresión evaluada con un conjunto de prueba de N elementos, donde la predicción del elemento i se representará con $\hat{y}_i$, e y_i es el valor real para dicho elemento:

[5]En este contexto, la distancia equivale a un valor que da información de cercanía o similaridad.

- **Error medio absoluto (MAE):** Proporciona un promedio del valor absoluto de los errores para todo el conjunto de prueba y viene dada por la siguiente ecuación:

$$MAE = \frac{1}{N} \cdot \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{2.24}$$

  Como se muestra en la ecuación, el aporte de cada error es lineal con respecto a la diferencia entre la predicción y el valor real.

- **Median Absolute Error (MedAE):** En comparación con la anterior, esta métrica brinda robustez ante la presencia de predicciones sobre instancias atípicas, lo cual puede resultar útil cuando se conoce a priori que las predicciones reales se encuentran cerca de un valor medio:

$$MedAE = mediana(|y_1 - \hat{y}_1|, ..., |y_N - \hat{y}_N|) \tag{2.25}$$

- **Error medio cuadrático (MSE):** Proporciona un promedio de los errores cuadráticos para todo el conjunto de prueba:

$$MSE = \frac{1}{N} \cdot \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{2.26}$$

  El aporte de cada error es cuadrático con respecto a la diferencia entre la predicción y el valor real, lo que significa que las diferencias pequeñas aportan menos que las diferencias grandes debido a la acción del exponente.

2.7.5. Modelos clásicos

Aunque en la actualidad hay cientos de modelos disponibles para ser usados en entornos de Aprendizaje Automático, queremos explicitar aquí aquellos que sirven de base a muchos de los desarrollados con éxito en los últimos tiempos y que serán utilizados de forma efectiva en los capítulos posteriores de esta memoria.

No daremos una presentación exhaustiva de los mismos, sino una primera aproximación que sirva para contextualizar y fijar las ideas principales que serán desarrolladas en los capítulos posteriores. Para el lector que lo desee, se proporcionan referencias bibliográficas que muestran con mayor detalle las características propias de cada uno de ellos.

Árboles de decisión

Los orígenes de los árboles de decisión datan del año 1958 (por ejemplo [7]). Sin embargo, no es hasta la aparición del trabajo de Leo Breiman *et al.*, *"Classification and Regression Trees"*[9], que se posicionaron como una herramienta importante dentro del Aprendizaje Automático.

Este modelo se compone de un conjunto de evaluaciones que se aplican a una instancia representada por un vector de características. Las evaluaciones corresponden a preguntas simples sobre los valores que pueden tomar elementos específicos del vector de características. Por ejemplo, si se tiene un conjunto de datos en donde sus instancias están compuestas por dos características, $\{C_1, C_2\}$, que pueden tomar los valores categóricos, $\{Bajo, Alto\}$, las posibles preguntas de evaluación del modelo pueden ser:

- ¿Es $C_1 = Bajo$?
- ¿Es $C_1 = Alto$?
- ¿Es $C_2 = Bajo$?
- ¿Es $C_2 = Alto$?

La aplicación de las evaluaciones en un árbol de decisión debe verificar ciertas restricciones:

- Las evaluaciones siguen un orden jerárquico ordenado acorde a la importancia de la pregunta, por tanto se evalúa la instancia con una pregunta a la vez.
- Dependiendo de la respuesta obtenida en cada evaluación se realiza la pregunta siguiente.
- No se puede repetir la misma pregunta (por ejemplo, ¿Es $C_1 = Bajo$? seguida por ¿Es $C_1 = Bajo$?).
- En el modelo no pueden existir preguntas completamente antagónicas sobre la misma característica (por ejemplo, ¿Es $C_1 = Bajo$? seguida por ¿Es $C_1 = Alto$?).

Con respecto a la primera regla, la definición del orden jerárquico de las preguntas se obtiene por la aplicación de un algoritmo que es capaz de crear de forma incremental la arquitectura del modelo[6]. Nótese que precisamente por el orden jerárquico de preguntas que dependen de la evaluación anterior se forma un "árbol"de decisiones consecutivas, que es lo que da nombre al modelo.

El algoritmo más famoso para la construcción automática de árboles de decisión a partir de conjuntos de datos es el algoritmo ID3 propuesto por John Ross Quinlan en su trabajo *"Induction of Decision Trees"*[82]. Esta propuesta encuentra cada pregunta del árbol de forma incremental por medio de una evaluación completa de todas las preguntas posibles acorde a un criterio de ganancia de información. En la Sección 3.2 se darán más detalles acerca de este modelo.

[6] Aquí la arquitectura del modelo viene determinada por las preguntas y su jerarquía.

Redes neuronales

Otro de los modelos de aprendizaje automático tan popular como antiguo son las redes neuronales artificiales. El primer estudio acerca de un modelo matemático que imite la forma en que cambian las señales en el cerebro humano a partir de la información de su entorno fue propuesto por Warren McCulloch y Walter Pitts en *"A logical calculus of the ideas immanent in nervous activity"*[70]. Sin embargo, no es hasta la aparición del trabajo de Bernard Widrow y Marcian Hoff *"Adaptive Switching Circuits"*[103] en 1960, que se propuso un modelo funcional llamado *ADALINE* y su versión extendida *MADALINE* y se implementan para aplicaciones reales de reconocimiento de patrones. Estos modelos en principio no podían ser escalados por la ausencia de sistemas de cómputo potentes en la época, pero tras la aparición y generalización de la arquitectura de von Neumann se presentaron más avances en esta área y las aplicaciones que podían resolver.

Las arquitecturas de redes neuronales artificiales propuestas hasta ese momento, y durante muchos años más, permitían el ingreso ponderado de los datos a ser procesados directamente a las unidades de cómputo que llamaron neuronas, desde donde posteriormente se obtenían los resultados. La ponderación es "aprendida" a partir de un conjunto de datos con el objetivo de minimizar el error existente entre la salida deseada y la salida estimada de la red por medio del algoritmo de *mínimos cuadrados*[7] (LMS, *Least Mean Square*, por sus siglas en inglés) propuesto por Widrow y Hoff. Cabe recalcar que en este tipo de redes los únicos parámetros que se aprenden son los pesos de entrada a la red, sin embargo las arquitecturas podían ser complementadas con operaciones de toma de decisión[8] u otras, pero con la restricción de tener parámetros fijos.

Como detalla el libro de Minsky y Papert *"Perceptrons: An Introduction to Computational Geometry"*[72], ADALINE con una sola capa de conexión (y las arquitecturas derivadas de este modelo) solamente pueden resolver problemas de aprendizaje de patrones linealmente separables con respecto a sus entradas. Sin embargo, para la fecha de publicación del libro, Rosenblatt ya había presentado un nuevo modelo multicapa denominado perceptrón multicapa en *"Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms"*[85] en 1962.

En este nuevo modelo existen 3 capas predefinidas: de entrada, oculta y de salida. Las conexiones entre la capa de entrada y la capa oculta es la misma que una red ADALINE, es decir se ponderan las entradas con un peso asociado a cada una de ellas antes de entrar a las neuronas, en este caso, de la capa oculta. Adicionalmente, se conectan las salidas de las neuronas de la capa oculta con las entradas de la capa de salida usando otro grupo de pesos. En este caso, los dos grupos de pesos deben ser aprendidos siguiendo algún criterio de minimización del error entre la salida de

[7] Este algoritmo se encuentra en la literatura con diferentes nombres: mínimos cuadrados, algoritmo Widrow-Hoff, regla delta de actualización, gradiente descendente para neuronas lineales, etc.

[8] En el modelo original de ADALINE se incluye una función de umbral a la salida de las neuronas para la toma de decisiones binarias.

la red y la salida deseada, pero hasta el momento no se conocía ningún algoritmo de minimización que diese ciertas garantías de éxito.

Una propuesta eficiente para el entrenamiento supervisado de este tipo redes no aparece hasta el año 1974 con la tesis doctoral de Paul Werbos, *"Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences"*[101], aunque no es hasta el año 1986 que se reinventa el algoritmo de Werbos en el trabajo de Rumelhart *et al. "Learning representations by back-propagating errors"* y se populariza el nombre del algoritmo de retropropagación del error, o *Backpropagation* por su nombre en inglés[9].

El algoritmo de Backpropagation marcó un antes y un después en el campo de las redes neuronales, convirtiéndose en la piedra angular para la resolución de un sinnúmero de tareas. Su objetivo es, dada una arquitectura de red multicapa, calcular los gradientes del error con respecto a cada parámetro (peso de la red), algo que logra mediante la aplicación sucesiva de la regla de la cadena para encontrar los gradientes de capas anteriores. Una explicación más detallada del modelo de red neuronal y su entrenamiento se presenta más adelante en la Subsección 4.2.2.

Máquinas de soporte vectorial (SVM)

Otro modelo de aprendizaje que para el día de hoy se puede considerar clásico son las *máquinas de soporte vectorial* (SVM, por sus siglas en inglés). Su aparición se dió en el trabajo de Boser, Guyon y Vapnik *"A training algorithm for optimal margin classifiers"*[8] en el año 1992, aunque las primeras ideas sobre el trabajo con vectores de soporte fueron presentadas antes en el artículo de Vapnik y Lerner, *"Pattern Recognition using Generalized Portrait Method"*[96], en 1963.

Su funcionamiento tiene como base la representación geométrica del conjunto de datos por la proyección de cada instancia del dataset como un punto en un espacio de igual dimensión al número de características existentes. Cada punto en este espacio es identificado por la clase a la que pertenece, pudiendo fundamentalmente existir a lo sumo dos clases para el uso de SVM[10]. Entonces, el problema de clasificación binaria se consiste en encontrar el "mejor"hiperplano que separe las instancias de las dos clases.

Como el número de hiperplanos que pueden separar el conjunto de puntos puede ser infinito, en SVM el término "mejor"hace referencia a un hiperplano que cumpla las siguientes dos condiciones:

- Minimiza el número de instancias mal clasificadas.

[9] En esta memoria usaremos tanto el nombre en español como en inglés al ser los dos de uso habitual en la comunidad científica.

[10] Un problema multiclase puede ser resuelto transformándolo a un conjunto de problemas de clasificación binaria.

- Maximiza el margen, o distancia existente, entre el hiperplano y las instancias más cercanas a este, las cuales se denominan *vectores de soporte*.

En consecuencia, SVM encuentra la mejor función lineal divisora para un conjunto de datos, que fue el aporte dado en [96]. Pero, ¿qué sucede si los conjuntos de puntos no son linealmente separables? En este caso las SVM utilizan el *truco del kernel* propuesto para el algoritmo de función potencial en [61]. Este truco utiliza una función, o *kernel*, para crear nuevas dimensiones a partir de las características existentes. La idea es que al usar una dimensión superior los datos sean linealmente separables. La separabilidad lineal en una dimensión superior equivale a la creación de una hipersuperficie en el espacio original, algo que fue explorado en el artículo de Boser, Guyon y Vapnik[8].

K-Nearest Neighbor (KNN)

Quizás el algoritmo de aprendizaje más simple y conocido es *K-Vecinos Cercanos* (KNN, por sus siglas en inglés). El artículo que propone el algoritmo tal como se conoce en la actualidad es *"Nearest neighbor pattern classification"*[21] de Cover y Hart, pero la idea se puede encontrar anteriormente en libros como *"Learning Machines: Foundations of Trainable Pattern-Classifying Systems"*[73] de Nils Nilsson y *"Decision-making processes in pattern recognition"*[90] de George Sebestyen. Incluso, según la revisión histórica de Pelillo en [81], la idea en la que se basa el algoritmo fue propuesta por Alhazen, un científico que vivió en los años 965-1040, que enunció que las características que determinan los individuos de un cierto tipo son idénticas.

El algoritmo de KNN puede ser definido por los siguientes pasos:

1. Al igual que las SVM, primero se proyectan las instancias del conjunto de datos como puntos de un espacio de tantas dimensiones como características, donde las instancias de cada clase son identificadas por su etiqueta.
2. También se proyecta la nueva instancia que requiere ser clasificada.
3. Basado en algún criterio de distancia predefinido, se eligen las K instancias más cercanas al punto que se desea clasificar.
4. De las K instancias más cercanas (vecinos) seleccionadas se cuenta el número que hay de cada clase.
5. Se clasifica la nueva instancia en la clase con mayor número de vecinos seleccionados.

Este algoritmo explora todo el conocimiento almacenado en el conjunto de entrenamiento para determinar cuál será la clase a la que pertenece una nueva muestra, pero únicamente tiene en cuenta los vecinos más próximos a ella, por lo que es lógico pensar que es posible que no se esté aprovechando de forma eficiente toda la información que se podría extraer del conjunto de entrenamiento.

En problemas prácticos donde se aplica este algoritmo de clasificación se acostumbra tomar un número K de vecinos impar para evitar posibles empates (aunque esta decisión solo resuelve el problema en clasificaciones binarias). En otras ocasiones, en caso de empate, se selecciona la clase que verifique que sus representantes tengan la menor distancia media al ejemplo que se está clasificando. En última instancia, si se produce un empate, siempre se puede decidir aleatoriamente entre las clases con mayor representación.

Una posible variante de este algoritmo consiste en ponderar la contribución de cada vecino de acuerdo a la distancia entre él y el ejemplar a ser clasificado, dando mayor peso a los vecinos más cercanos. Esta mejora es muy efectiva en muchos problemas prácticos, proporcionando un algoritmo robusto ante ruido en los datos y suficientemente efectivo en conjuntos de datos grandes. Además, se puede ver que al tomar promedios ponderados de los K vecinos más cercanos el algoritmo puede evitar el impacto de ejemplos con ruido aislados.

CAPÍTULO 3

EXTRACCIÓN DE CARACTERÍSTICAS Y CLASIFICACIÓN CON UN ENFOQUE CLÁSICO

3.1. Introducción

Hay un incremento constante de requerimientos para el trabajo continuo de cierto sistemas dinámicos como son las máquinas de transmisión de potencia. Esta es la razón del por qué son tan valorados los nuevos enfoques para la creación de sistemas de diagnóstico de fallo precisos y confiables. Actualmente existen estudios invaluables para detectar cambios en los estados de distintos componentes de un sistema dinámico mediante el uso de técnicas estándar de diagnóstico, como son Cepstrum o análisis de envolvente con la transformada de Hilbert [84]. Todos estos esfuerzos se han enfocado en aplicaciones concretas del reconocimiento de estados de sistemas dinámicos, en algunos casos aplicados al diagnóstico de fallos en elementos de maquinaria rotativa.

En los años recientes varias técnicas de análisis para el diagnóstico de fallas en maquinaria rotativa han usado la Transformada de Wavelet Packet (WPT) con el fin de resaltar la información que comúnmente proporcionan los parámetros estadísticos clásicos de la señal capturada del sistema dinámico en los dominios de tiempo y frecuencia [104, 45, 64].

En el caso del diagnóstico basado en Machine Learning, los enfoques más comunes se han desarrollado usando Redes Neuronales [66, 58], Máquinas de Soporte Vectorial [33], Análisis de Clusters y Algoritmos Genéticos (GA) [40, 87]. Estos enfoques han sido muy útiles en su aplicación al mantenimiento basado en la condición,

como se presenta en Jardine et al. [42].

Recientemente, el modelo de aprendizaje automático llamado Random Forest (RF), basado en árboles de decisión, se ha usado como técnica de clasificación para el diagnóstico de fallas en múltiples áreas de la ingeniería por su robustez ante la presencia de un gran número de características de entrada, un número limitado de instancias disponibles para el aprendizaje, y por la alta interpretabilidad de los modelos basados en árboles [60, 97]. En el diagnóstico de la condición de maquinaria, este modelo se ha usado con algoritmos genéticos con el fin de mejorar la exactitud en la clasificación [105, 46]. Yang et al. En [105] los autores usan GA para optimizar dos parámetros del algoritmo de RF: el número de árboles y el número de variables divisoras en cada nodo de los árboles. En motores de combustión, Karabadji et al. [46] aplican GA para seleccionar el mejor clasificador basado en árboles para el diagnóstico de fallas en ventiladores industriales, el objetivo del análisis fue encontrar el clasificador que aparece más frecuentemente en la población.

Por otro lado, los parámetros de la condición para el diagnóstico de fallas están principalmente relacionados con mediciones estadísticas obtenidas de las señales en el dominio de tiempo y frecuencia. Además, los parámetros asociados a este dominio tienen información importante acerca de la condición de la máquina, y son también usados con el fin de extender el conjunto de parámetros de la condición que a su vez son procesados en los algoritmos de diagnóstico. Como hemos comentado en capítulos anteriores, estos parámetros de la condición son los que conocemos como *características* en tareas de aprendizaje automático. Tomando en cuenta la disponibilidad de un gran número de características candidatas para el diagnóstico de fallas, el problema de su selección óptima tras el proceso de extracción permanece todavía abierto. Debido a que este problema puede ser interpretado como una tarea de optimización, del que en general se desconoce una formalización analítica, es normal que haya sido abordado con GA, pero es bien conocido que el proceso de optimización con GA es computacionalmente costoso, por lo que requiere un tiempo excesivamente grande de procesamiento para obtener convergencia a una respuesta aceptable.

Es por ello que comenzamos explorando la capacidad de RF como un mecanismo de selección de características, así como clasificador con un número de instancias limitadas dentro de una metodología para la identificación de estados de un sistema dinámico que puede ser usada en aplicaciones de diagnóstico de fallos. Así pues, y con el fin de comprender correctamente cómo funciona el modelo de Random Forest, comenzaremos profundizando en el modelo básico sobre el que se soporta, los árboles de decisión.

3.2. Árboles de decisión

Como vimos en el capítulo anterior, los modelos basados en árboles[7] se sitúan entre los más populares dentro del aprendizaje automático [28]. Esta popularidad se

debe a la capacidad expresiva que muestran al representar conceptos que definen a un problema determinado, lo que permite que el modelo pueda ser fácilmente interpretado por un humano e incluso pueda ser traducido a expresiones lógicas de forma sencilla.

3.2.1. Arquitectura de un árbol de decisión

Muchos problemas complejos se pueden resolver mediante la aplicación de un conjunto de preguntas de forma iterativa. Estas preguntas trabajan como filtros que discriminan la información ingresada, de forma que la aplicación de un filtro depende de la respuesta dada al filtro anterior. A medida que se aplican estos filtros el espacio de posibles respuestas se hace cada vez más pequeño y, en consecuencia, tendremos una mayor certeza en la respuesta entregada ante una entrada determinada.

Todo este proceso puede ser representado de manera gráfica mediante un árbol (véase Figura 3.1) que consta de un nodo inicial, que se denomina comúnmente *nodo raíz*, en donde se realiza la primera pregunta, un conjunto de *nodos internos* (también llamados nodos de división), y un conjunto de *nodos hoja o terminales*, que es en donde se obtiene el resultado. Aunque puede haber una cantidad indeterminada de nodos *hijos*, es muy habitual trabajar con árboles binarios (aquellos en los que cada nodo tiene, a lo más, 2 hijos), ya que es posible convertir cualquier árbol en un árbol binario equivalente (equivalente respecto a las respuestas que proporciona).

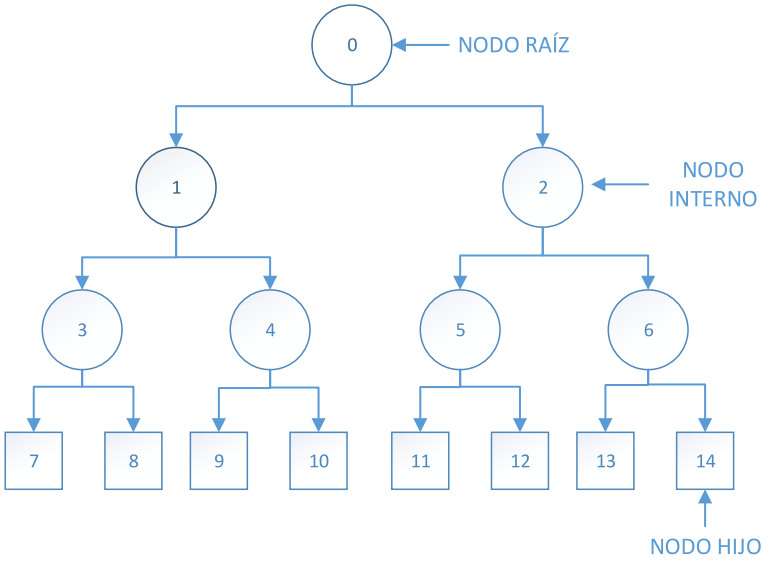


Figura 3.1: Estructura de un árbol de decisión.

Ilustremos de forma sencilla la interpretación y construcción de los árboles de decisión. Supongamos el proceso de determinar si una persona es “Sedentaria” o “Activa”. Para ello se ha elaborado una encuesta con la pregunta inicial “¿Hace de-

porte?". Si la respuesta es "No" inmediatamente se concluye "Sedentaria". Caso contrario, si la respuesta es "Si", se continúa con la siguiente pregunta, "¿Lo realiza por mas de 2 horas semanales?". Si la respuesta es "No" se concluye "Sedentaria". Caso contrario, si la respuesta es "Si", se concluye "Activa". El árbol de decisión resultante se podría representar como muestra la Figura 3.2.

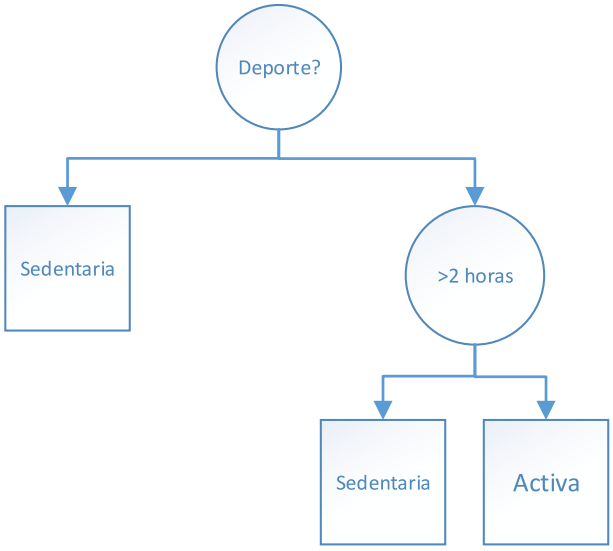


Figura 3.2: Estructura del árbol generado para determinar si una persona es sedentaria o activa.

3.2.2. Formalización

Anteriormente se mencionó que se llama árbol de decisión a un conjunto de preguntas organizadas de forma jerárquica en una estructura llamada árbol. Cada pregunta se asocia a un nodo del árbol y formalmente se representa mediante una función de división de la forma:

$$h(D_i, \theta_j) : \mathbb{R}^n \times T \rightarrow \{0, 1\} \tag{3.1}$$

Esta función muestra que, para un nodo j, una instancia específica D_i es clasificada como 0 o 1 dependiendo de las características de la instancia y los parámetros de división θ_j asociados a ese nodo, representados por T. Dependiendo del resultado devuelto por la función de división ($\{0, 1\}$, $\{No, Si\}$, $\{Falso, Verdadero\}$) el vector de características D_i descenderá hacia el nodo de la izquierda o al de la derecha según corresponda. Así, desde la perspectiva de los modelos de aprendizaje automático, un árbol de decisión es un conjunto de modelos de clasificación simples apilados en una estructura jerárquica para construir un modelo con mejores propiedades.

El conjunto de entrenamiento inicial (y que podrá ser obtenido por cualquiera de los procesos de división del conjunto de datos mostrados en el capítulo de Funda-

mentos) se denotará como S_0, donde el subíndice indica que ingresa al nodo raíz. En general, los conjuntos de entrenamiento que ingresarán en los nodos hijo Izquierdo y Derecho de un nodo padre j serán S_j^I y S_j^D, respectivamente (véase Figura 3.3), y deben cumplir las siguientes propiedades:

$$S_j = S_j^D \cup S_j^D,\ S_j^I \cap S_j^D = \varnothing \tag{3.2}$$

$$S_j^I = S_{2j+1},\ S_j^D = S_{2j+2} \tag{3.3}$$

La ecuación (3.2) hace referencia a que no es posible excluir ni se puede perder ninguna instancia del conjunto S_j durante el procesamiento que realiza el nodo j, y no es posible que las mismas instancias de entrenamiento, al ser evaluadas en la función de división del nodo j, puedan obtener resultados distintos. La ecuación (3.3) hacen referencia a la convención utilizada para nombrar a cada nodo en orden numérico, de izquierda a derecha, y de arriba a abajo.

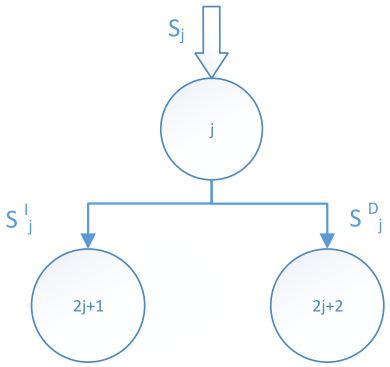


Figura 3.3: Estructura de nodo padre e hijos y su numeración correspondiente.

3.2.3. Entrenamiento de un árbol de decisión

El éxito o fracaso de los modelos basados en árboles está definido por la capacidad que tenga cada nodo de clasificar una instancia de manera correcta hacia la izquierda o derecha según corresponda. En consecuencia, es posible definir la fase de entrenamiento como el proceso por el cual, a partir de un conjunto de entrenamiento inicial S_0, se encuentran los parámetros de división adecuados para cada nodo (*aprendizaje de parámetros*). En este tipo de modelos, tras haber definido cada nodo en la fase de entrenamiento, no es posible volver a un nodo anterior para realizar ningún cambio, de ahí la importancia en la elección correcta.

El aprendizaje de parámetros es una búsqueda en el espacio de parámetros T, qu en general no puede ser resuelta por técnicas voraces debido a la cantidad de soluciones candidato que son coherentes con los datos. Por ejemplo, para un problema con n características que pueden tomar valores binarios, la cantidad de posibles soluciones que se requeriría probar es 2^{2^n}. Por esta razón, en los árboles de decisión esta

búsqueda se realiza en cada nodo y su criterio principal es el de la optimización de una función objetivo que puede representarse como la selección de parámetros que "mejor" logre dividir el conjunto S_j en los conjuntos S_j^I y S_j^D. La función objetivo es la que definirá el concepto de "mejor" según sea la aplicación.

Se han diseñado varios algoritmos para atacar el problema de búsqueda en la fase de entrenamiento (el más famoso es el algoritmo ID3 en [82]). La diferencia principal entre ellos radica en la función objetivo que utilizan para garantizar una separación en los conjuntos resultantes.

El problema de optimización general que hay que resolver puede ser representado por:

$$\theta_j = arg\ \underset{\theta \in T}{max} I(S_j, \theta) \tag{3.4}$$

que expresa que los parámetros de división θ_j seleccionados serán aquellos del conjunto de parámetros de división, T, que maximicen la función objetivo I en el nodo j sobre el conjunto de entrenamiento S_j. La función objetivo I puede ser cualquier criterio que logre expresar una medida de "pureza" de la información entregada por el conjunto de parámetros con respecto al conjunto de entrenamiento, medida comúnmente llamada *ganancia de la información*. Entre las funciones más utilizadas para el cálculo de la ganancia de información están la *entropía* y el *mejor divisor Gini*.

Entropía

Es necesario primero definir el concepto de entropía en este contexto como una medida de pureza de la información implícita en un conjunto de entrenamiento S. En árboles de decisión para problemas con c clases se utiliza la formalización matemática de la entropía c-aria de Shannon.

Sea C el conjunto de valores que puede tomar la variable objetivo en el conjunto S_j:

$$C = \{c_1, c_2, ..., c_c\} \tag{3.5}$$

y su distribución de probabilidad asociada (es decir, la probabilidad de ocurrencia para cada una de las clases anteriores):

$$P = \{p_1, p_2, ..., p_c\} \tag{3.6}$$

donde la probabilidad de la clase c_i en S_j es:

$$p_i = \frac{n_{c_i}}{n_{S_j}} \tag{3.7}$$

siendo n_{c_i} el número de instancias de entrenamiento de clase c_i y n_{S_j} el número de elementos total en el conjunto S_j.

Entonces, la entropía c-aria de S_j vendrá dada por:

$$H(S_j) = -\sum_{l=1}^{c} p_l \cdot log_c p_l \tag{3.8}$$

En la Figura 3.4 se muestra una gráfica de entropía 2-aria en función de la probabilidad $p = p_1$ (ya que, en este caso, p_2 está determinada y vale $1 - p_1$). En este caso sencillo es fácil interpretar cómo la entropía mide el grado de dispersión y la ausencia de homogeneidad en un conjunto dado. Para la entropía 2-aria, usada para evaluar características binarias (usadas para los árboles de decisión binarios), se aprecia que el mayor valor se presenta para una probabilidad de $p_1 = p_2 = 0{,}5$, lo que en un proceso estocástico equivaldría a una máxima incertidumbre. Esta idea

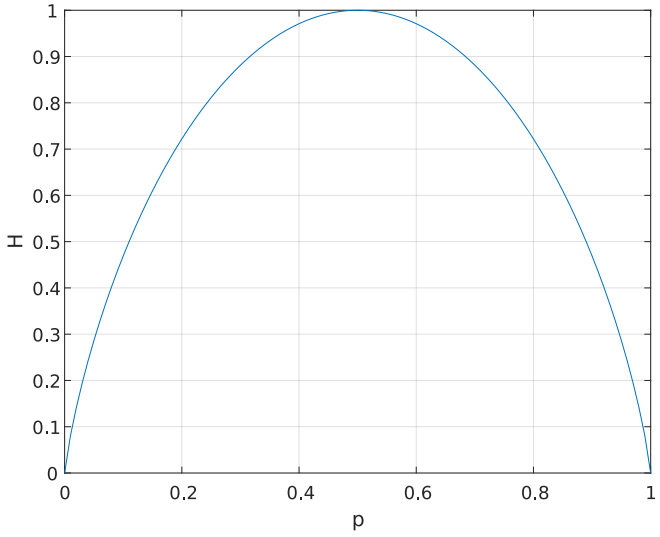


Figura 3.4: Entropía 2-aria en función de $p = p_1$.

intuitiva sigue siendo cierta para la entropía c-aria, obteniendo un máximo de entropía cuando $p_1 = \cdots = p_c = \frac{1}{c}$ y valor mínimo (nulo) cuando hay certidumbre y alguna de las probabilidades $p_i = 1$.

Partiendo de la definición de entropía dada previamente, definimos la ganancia de información como una medida de variación de la entropía tras utilizar una de las características posibles, A, para separar los ejemplos de un conjunto S_j que ingresan a un nodo:

$$I_{GANANCIA}(S_j, A) = H(S_j) - \sum_{i \in \{I,D\}} \frac{n_{S_j^i}}{n_{S_j}} H(S_j^i) \tag{3.9}$$

donde S_j^i es el subconjunto de S_j obtenido al dividirlo mediante A a la izquierda o derecha según corresponda.

Como heurística habitual en la construcción automática de un árbol de decisión, se busca obtener la característica que proporcione la mayor ganancia de información.

Mejor divisor Gini

Tomando como base la ecuación (3.5) y la ecuación (3.6) se puede introducir otro criterio de ganancia de información llamado índice Gini [83], que viene definido por:

$$G(S_j) = 1 - \sum_{p \in P} p^2 \tag{3.10}$$

Cuando todos los elementos de S_j pertenecen a la misma clase el índice Gini es igual a 0. Si la distribución de clases es equilibrada (existe igual número de instancias de todas las clases en S_j), el índice Gini es igual a $0{,}5$.

De forma equivalente al caso anterior, la ganancia de información para una característica A usando este índica, que es la función objetivo, resulta:

$$I_{GINI}(S_j, A) = G(S_j) - \sum_{i \in \{I,D\}} \frac{n_{S_j^i}}{n_{S_j}} G(S_j^i) \tag{3.11}$$

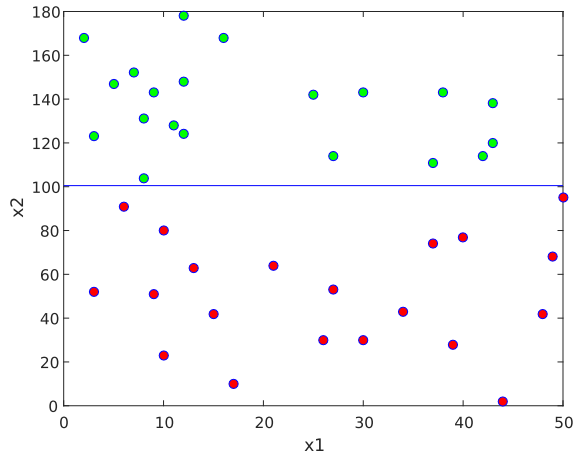
3.2.4. Representación gráfica

Cuando el problema que se aborda es definido por dos o tres características, se puede representar en el plano o espacio cartesiano respectivamente, y es posible mostrar gráficamente la acción que realiza una función de división en un nodo.

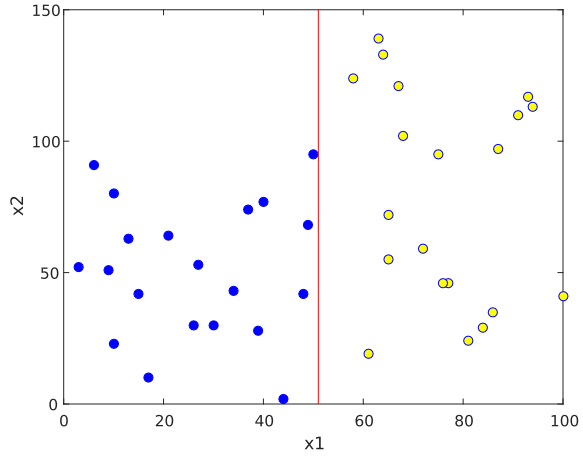
Normalmente, en los parámetros de división solamente se encuentra una característica divisora que permite discriminar los datos. Para dar un ejemplo, en la Figura 3.5 se presentan dos casos de divisor acorde a la característica seleccionada para una nodo específico usando el criterio de ganancia de información, donde se observa que el resultado de aplicar una sola característica divisora en cada nodo obtendrá un hiperplano divisor alineado siempre con la característica no considerada o, lo que es lo mismo, perpendicular a la característica divisora.

La orientación del hiperplano es elegida en la fase de entrenamiento de acuerdo a la ganancia de información obtenida al seleccionar una u otra característica para dividir los datos. El criterio de ganancia de información puede ser también representado de forma gráfica usando las masas de probabilidad iniciales (para S_j) y las masas de probabilidad posteriores (para los hijos S_j^I y S_j^D). Así se muestra en la Figura 3.6 un típico problema de clasificación multi-clase en donde se pueden apreciar cuatro clases, y, por su masa de probabilidad (Figura 3.6b) se sabe que todas sus clases tienen igual número de elementos.

Al aplicar la función de división $x_2 < 100{,}5$ (ver Figura 3.7a) al nodo raíz, no se logra dividir de manera correcta el conjunto inicial, obteniendo una probabilidad posterior en los nodos hijos que muestran instancias de clases que debieron ser discriminadas en el nodo padre (ver Figura 3.7b y Figura 3.7c). Este resultado se puede corroborar con un valor pequeño de la ganancia de información igual a $0{,}3522$.

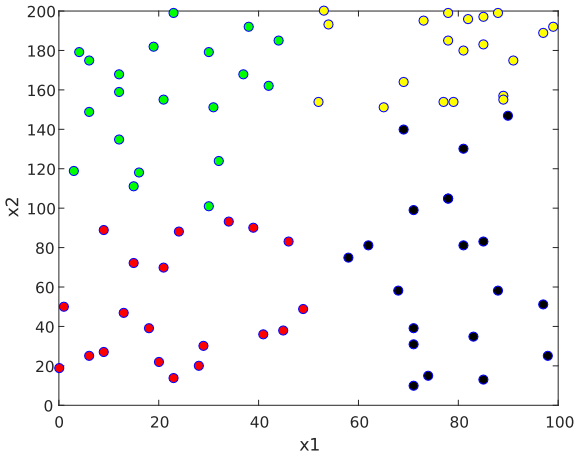


(a) Divisor horizontal (alineado con el eje x_1), la función de división correspondiente en el nodo sera "$x_2 < 100{,}5$".

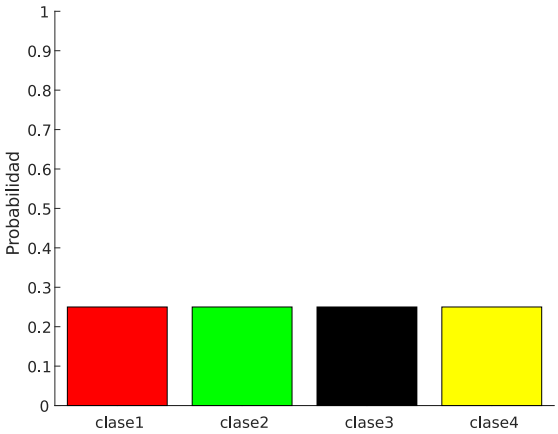


(b) Divisor vertical (alineado con el eje x_2). La función de división asociada al nodo es "$x_1 < 51$".

Figura 3.5: Representación en el espacio cartesiano de divisores alineados con el eje.

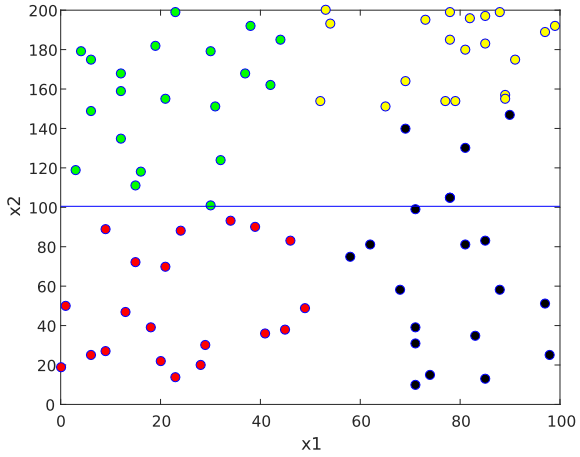


(a) Representación en el espacio cartesiano del problema. Cada color representa una clase diferente.

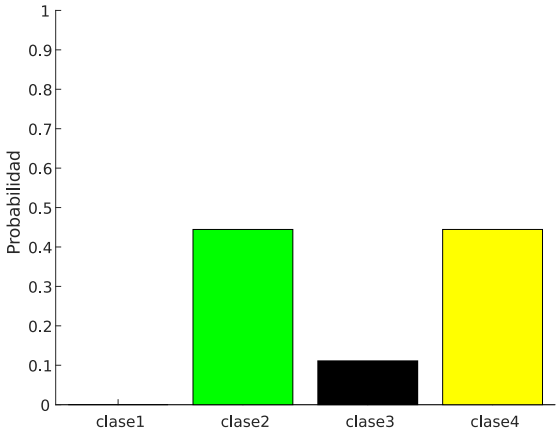


(b) Probabilidades iniciales del conjunto de entrenamiento.

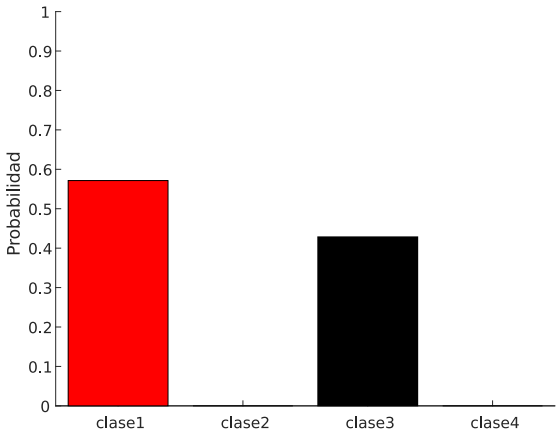
Figura 3.6: Problema de clasificación de cuatro clases.



(a) Representación de la aplicación del divisor horizontal alineado con x1. $I_{GANANCIA} = 0{,}3522$.



(b) Probabilidad posterior en el nodo hijo izquierdo.



(c) Probabilidad posterior en el nodo hijo derecho.

Figura 3.7: Problema de cuatro clases con divisor horizontal.

Por otra parte, si se aplica una función de división $x_1 < 51$ (ver Figura 3.8a), la división de las elementos en el conjunto es adecuada, lo que permite obtener subconjuntos de instancias en los nodos hijo con mayor pureza, como se muestra en la Figura 3.8b y Figura 3.8c.

3.2.5. Criterios de parada y podas

Un árbol de decisión puede continuar su crecimiento en profundidad mientras tenga instancias en el conjunto de entrenamiento que puedan ser divididas en clases y, adicionalmente, tenga características para formar la función de división. Tras ello, la generación de nuevos nodos internos no tendría ningún efecto positivo en la clasificación de los datos. Sin embargo, existen también criterios de parada temprana que, aunque todavía sea posible dividir el conjunto de datos, son capaces de detener la fase de entrenamiento. El más utilizado es el criterio de profundidad, que detiene la creación de nodos al sobrepasar un determinado nivel de profundidad del árbol (que ha sido prefijado).

Independientemente de la utilización de un criterio de parada específico, tras el aprendizaje de parámetros de todos los nodos es necesario definir la correspondencia de cada nodo hoja con una clase. Esto se realiza obteniendo la distribución de probabilidad en el nodo hoja y eligiendo como resultado para ese nodo la clase con mayor probabilidad. De esta forma, se puede tratar de manera separada el criterio de parada de las clases asignadas a los nodos hoja.

Además, también es posible la utilización de técnicas de poda, que simplifican la estructura del árbol eliminando divisiones intermedias y permiten evitar sobreentrenamiento e incrementar la generalidad del árbol.

3.2.6. Fase de Prueba

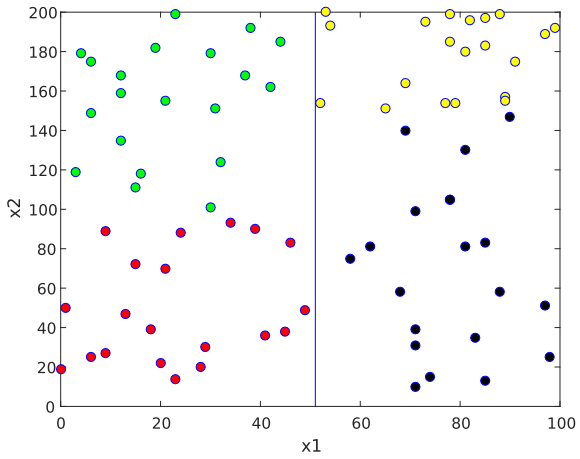
Tras la generación del árbol de decisión en la fase de entrenamiento, es posible clasificar una instancia nueva haciéndola descender por el árbol y sometiéndola a las pruebas en cada nodo (ver Figura 3.9), asignando como clasificación de la instancia la correspondiente al nodo hoja al que haya llegado.

3.2.7. Desventajas

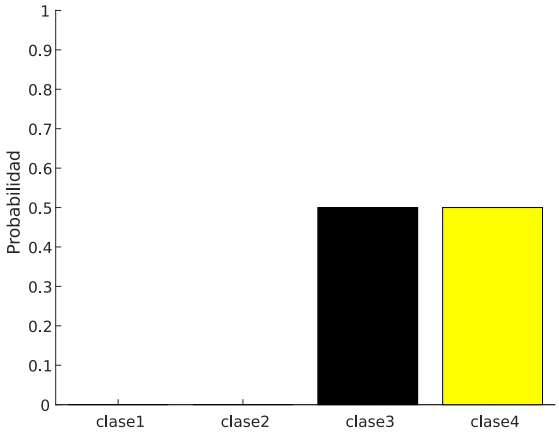
Si bien es cierto que el uso de árboles de decisión para resolver tareas de aprendizaje presenta grandes ventajas, sobre todo en la claridad de la presentación de los resultados, los árboles tienen bajas tasas de clasificación en comparación con otras técnicas como LDA y Redes Neuronales [23].

Entre las principales desventajas de este modelo están:

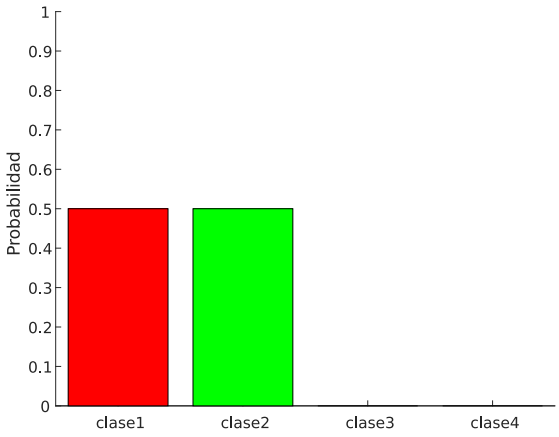
- Poca robustez ante la presencia de ruido en los ejemplos de entrenamiento.



(a) Representación de la aplicación del divisor vertical alineado con x2. "$I_{GANANCIA} = 0{,}5$".



(b) Probabilidad posterior en el nodo hijo izquierdo.



(c) Probabilidad posterior en el nodo hijo derecho.

Figura 3.8: Problema de cuatro clases con divisor vertical.

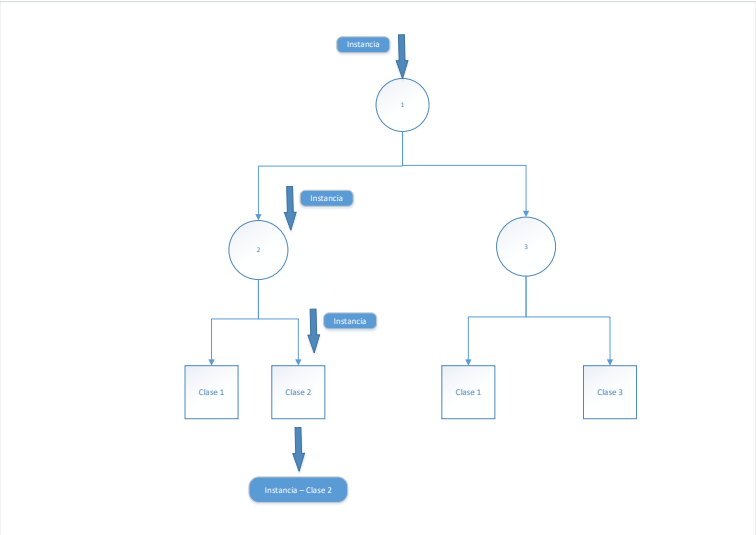


Figura 3.9: Ruta seguida por una instancia en una nueva clasificación posterior a la fase de entrenamiento.

- Poca eficiencia al lidiar con problemas de alta dimensión de características.
- Poca eficiencia al trabajar con un conjunto de entrenamiento demasiado grande.
- Uso de fases posteriores de poda para atacar el problema del sobre-ajuste.

3.3. Random Forest

Random Forest [10] es un modelo de aprendizaje automático que persigue tener las ventajas de los árboles de decisión agrupando una gran cantidad diversa de ellos. Cada árbol de decisión que es parte del modelo de Random Forest trabaja de forma independiente a los demás para luego finalmente utilizar alguna técnica de agregación de resultados y obtener una conclusión final.

A continuación se presenta la teoría y formalización de los *conjuntos de modelos*, que son la base en la que se fundamenta el modelo de Random Forest.

3.3.1. Conjuntos de modelos

Los conjuntos de modelos son en sí mismos modelos de aprendizaje automático capaces de realizar tareas de clasificación, regresión, etc. Los submodelos que los componen, llamados *aprendices débiles*, generalmente trabajan de forma independiente de los demás, contrariamente a lo que sucedía con los árboles de decisión

donde la salida de un nodo sirve de entrada de otro. De esta forma, un conjunto h de K modelos puede ser representado como:

$$h = \{h_1(\mathbf{x}), h_2(\mathbf{x}), ..., h_k(\mathbf{x}), ..., h_K(\mathbf{x})\} \tag{3.12}$$

siendo $\mathbf{x} \in \mathbb{R}^n$ una variable aleatoria que sigue la misma distribución X (comúnmente desconocida) que representa el conjunto de datos por muestreo (el conjunto de entrenamiento).

Los aprendices débiles entregan una respuesta independiente ante una instancia específica que requiere ser clasificada. Esto significa que los parámetros asociados a cada aprendiz también tiene independencia de los demás.

Por ejemplo, sin caer en una pérdida de generalidad, si los aprendices débiles pertenecen a una misma familia de modelos (árboles de decisión, por ejemplo) se puede escribir cada uno de ellos como:

$$h_k(\mathbf{x}) = h(\mathbf{x}|\theta_k) \tag{3.13}$$

siendo θ_k los parámetros asociados al aprendiz, por ejemplo: arquitectura, características de entrada, etc. Todo depende del modelo que define al aprendiz débil.

Al final, cuando se ha obtenido el resultado para una clase específica (un voto) con cada aprendiz, se agregan los votos y se entrega como respuesta final del conjunto de modelos, por ejemplo, la clase más votada o, de forma más general, en función de la probabilidad empírica, la cual para una clase específica c_i se obtiene de la siguiente forma:

$$\hat{P}(c_i|\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K} I(h(\mathbf{x}|\theta_k) = c_i) \tag{3.14}$$

donde la función $I(\cdot)$ devuelve 1 si su argumento es verdadero, y 0 en caso contrario. Finalmente, la clase elegida como resultado de h es aquella que tiene mayor probabilidad empírica.

3.3.2. Función marginal empírica y error de generalización

Una medida de confianza del modelo de clasificación por conjunto de modelos es la función marginal empírica, que devuelve la diferencia entre el promedio de aprendices que clasificaron la instancia correctamente y el promedio de aprendices que clasificaron la instancia como la segunda opción más probable:

$$\hat{mg}(\mathbf{x}, y) = \frac{1}{K}\sum_{k=1}^{K} I(h(\mathbf{x}|\theta_k) = y) - max_{j \neq y}\frac{1}{K}\sum_{k=1}^{K} I(h(\mathbf{x}|\theta_k) = j) \tag{3.15}$$

$$= \hat{P}_k(h_k(\mathbf{x}) = y) - max_{j \neq y}\hat{P}_k(h_k(\mathbf{x}) = j) \tag{3.16}$$

Si $\hat{mg}(\mathbf{x}, y) > 0$ entonces el conjunto de modelos realiza una correcta clasificación de $\mathbf{x}$. Si por el contrario $\hat{mg}(\mathbf{x}, y) < 0$, el conjunto de modelos realiza una clasificación incorrecta.

En este sentido, el error de generalización viene determinado por la probabilidad de que el conjunto de modelos realice una clasificación incorrecta de $\mathbf{x}$, es decir:

$$e = P_{\mathbf{x},y}(\hat{mg}(\mathbf{x}, y) < 0) \tag{3.17}$$

3.3.3. Convergencia en Random Forest

Como se mencionó anteriormente, Random Forest es un modelo creado a partir de un conjunto de modelos donde cada $h_k(\mathbf{x})$ es un árbol de decisión. Por tanto, en este caso θ_k define la estructura del k-ésimo árbol, así como también el conjunto de datos y las características a ser evaluadas en cada nodo para su creación. Estos dos últimos parámetros serán discutidos en profundidad más adelante.

En [11], Leo Breiman demuestra que si los aprendices débiles son árboles y el número de estos se incrementa, el error de generalización converge a una cota inferior determinada, es decir:

$$e \underset{K\to\infty}{\to} P_{\mathbf{x},y}\big[P_\theta(h(\mathbf{x}|\theta) = y) - max_{j\neq y} P_\theta(h(\mathbf{x}|\theta) = j) < 0\big] \tag{3.18}$$

Este resultado demuestra dos cosas importantes:

1. A pesar del incremento del número de aprendices, el modelo final no se sobreajusta a los datos.
2. No importa cuánto se incremente el número de árboles, no se puede disminuir el error más allá de una cierta cota.

Los detalles de la demostración de este resultado se muestran a continuación. Nótese que para demostrarlo es necesario y suficiente mostrar que:

$$\frac{1}{K}\sum_{k=1}^{K} I(h(\mathbf{x}|\theta_k) = j) = P_\theta(h(\mathbf{x}|\theta) = j) \tag{3.19}$$

Para ello, para un conjunto de datos fijo y un aprendiz con parámetros θ, obsérvese que en el espacio de características el conjunto de $\mathbf{x}$ tal que $h(\mathbf{x}|\theta) = j$ es la unión de hiper-rectángulos en $\mathbb{R}^n$, donde cada uno de ellos puede ser a su vez definido por una colección de intervalos $\{I_i\}_{i=1}^n$ de la siguiente manera:

$$R = \{\mathbf{x}|x_i \in I_i\} \tag{3.20}$$

Así, el conjunto de uniones de hiper-rectángulos donde cada k-ésima unión representa el espacio que el aprendiz k cataloga como j, puede ser escrito como $\{S_k\}_{k=1}^K$. A partir de lo anterior se puede definir la función $\phi(\theta) = k$ si $\{\mathbf{x} : h(\mathbf{x}|\theta) = j\} = S_k$

y el contador N_k como el número de veces que el espacio que define a un aprendiz en el espacio de características es igual a la k-ésima unión de hiper-rectángulos, es decir N_k es el número de veces que $\phi(\theta_m) = S_k$. Así, N_k viene dado por:

$$N_k = \sum_{m=1}^{M} I(\phi(\theta_m) = S_k) \tag{3.21}$$

Entonces, usando las funciones definidas anteriormente, el lado izquierdo de la ecuación (3.19) se puede reescribir como:

$$\frac{1}{M}\sum_{m=1}^{M} I(h(\mathbf{x}|\theta_m) = j) = \frac{1}{M}\sum_{k=1}^{K} N_k I(\mathbf{x} \in S_k) \tag{3.22}$$

Luego, por la ley de los grandes números $\frac{1}{M}N_k$ converge a $P_\theta(\phi(\theta) = k)$ que, reemplazado en la ecuación (3.22), concluye:

$$\frac{1}{M}\sum_{k=1}^{K} N_k I(\mathbf{x} \in S_k) \to \sum_{k=1}^{K} P_\theta(\phi(\theta) = k) I(\mathbf{x} \in S_k) \tag{3.23}$$

$$\to P_\theta(h(\mathbf{x}|\theta) = j) \tag{3.24}$$

que es lo que se quería demostrar.

3.3.4. Factores determinantes

Otro resultado que Leo Breiman encontró, tomando como base el trabajo de Amit y Geman[6], es una cota superior para el error de generalización en conjuntos de modelos, la cual es dependiente de dos factores fundamentales: la fuerza del conjunto de modelos, y la correlación entre los aprendices débiles. Su hallazgo es como sigue:

$$e \leq \hat{\rho}\frac{(1 - s^2)}{s^2} \tag{3.25}$$

donde $\hat{\rho}$ es el promedio de las correlaciones $\rho(\theta, \theta')$ para todos los posibles pares de aprendices (θ, θ') en el conjunto de modelos, y el término s es una medida de fuerza de los aprendices débiles, que viene dada por:

$$s = E_{\mathbf{x},y}(\hat{mg}(\mathbf{x}, y)) \tag{3.26}$$

En la ecuación (3.25) se puede apreciar que es deseable tener un correlación pequeña entre los aprendices débiles, es decir que exista diversidad entre ellos para que la cota de error disminuya. Al mismo tiempo, es deseable que la fuerza de los aprendices sea cercana a 1, lo cual se logra, según la ecuación (3.26), con aprendices que en general se equivoquen en muy pocas ocasiones. A partir de este resultado, Breiman propone el modelo Random Forest, donde el problema de la correlación entre árboles es atacado inyectando aleatoriedad al sistema. En este modelo se han implementado dos técnicas para la inclusión de componentes aleatorias, el *bagging* y la *selección aleatoria de características*, que detallamos a continuación.

3.3.5. Bagging

Su nombre viene de "Bootstrap aggregating", y es la aplicación del método de bootstrap [29] para la creación de árboles diversos a partir de distintos conjuntos de entrenamiento para, posteriormente, combinar sus funciones y obtener un modelo más fuerte. En el bagging se realiza un submuestreo del conjunto de entrenamiento, S_0, para obtener K nuevos subconjuntos mediante el método de bootstrap [10, 26]. Para formar estos subconjuntos esta técnica de muestreo toma N ejemplos de S_0 con reemplazo, lo que significa que, tras tomar cada elemento, éste no desaparece del conjunto inicial, teniendo así la misma probabilidad de tomar nuevamente cada instancia a lo largo del proceso.

Cada subconjunto entrena un árbol de decisión que por sí mismo es capaz de predecir (de manera correcta o incorrecta) una nueva instancia. Los resultados de las predicciones que cada modelo ha obtenido son agregados (unidos) por votación del conjunto de modelos. La clase que se entrega como resultado es la que ha obtenido mayoría de votos.

El Bagging no se encuentra limitado a su uso con árboles, y puede ser aplicado con cualquier modelo de aprendizaje manteniendo los mismos principios.

Usando la formalización anterior, para un aprendiz $h_k(\mathbf{x})$ caracterizado por el vector de parámetros θ_k, aplicar el proceso de bagging equivale a tener en el elemento θ_k^b de θ_k (que contiene los indices de las instancias de $\mathbf{D}$ con los que se entrenará el modelo) un conjunto de números aleatorios generados con el proceso de bootstrap descrito anteriormente.

Un concepto importante que aparece con el uso de este método es el *Out Of Bag Error* (*oob-error*), que se define como el valor promedio de todos los errores cometidos por cada aprendiz débil evaluados en el subconjunto de datos que no fue usado para su creación, y que proporciona una primera métrica no sesgada de *accuracy* para los modelos basados en colecciones utilizando solamente el conjunto de entrenamiento.

3.3.6. Selección aleatoria de características

La selección aleatoria de características también ataca el problema de correlación entre los árboles del Random Forest, mejorando notablemente la eficiencia del modelo en la fase de entrenamiento. Como se observa en la ecuación (3.4), la búsqueda realizada en cada nodo para maximizar la función se efectúa en el conjunto T completo [22].

Existen varios casos en los que la medida de T tiende a infinito, haciendo que la búsqueda no pueda converger en un tiempo finito, e imposibilitando totalmente la resolución del problema. Para evitar este problema, en Random Forest se reformula la ecuación (3.4) de la siguiente manera:

$$\theta_j = \underset{\theta \in T_j}{arg\ max} I(S_j, \theta) \tag{3.27}$$

siendo $|T_j| \ll |T|$, en dónde T_j es un submuestreo aleatorio de T.

Normalmente la elección de la medida de T_j es igual para todos los nodos en todos los árboles, lo que permite insertar un parámetro general $\rho = |T_j|$ común a todo el conjunto. Si ρ tiende a $|T|$ entonces la correlación aumenta por la ausencia de aleatoriedad en el sistema. Si ρ tiende a 1 sucede lo contrario, y se elimina totalmente el proceso de optimización en cada nodo al existir solamente un característica asignada de forma aleatoria en el mismo.

3.4. Metodología para la clasificación de estados con un enfoque totalmente supervisado

Tomando como base los conceptos presentados anteriormente, en esta sección se introduce una metodología para la clasificación de estados de una variable en un sistema dinámico. El enfoque utilizado para la generación del modelo es el enfoque clásico que, adaptado para trabajar con series obtenidas desde el sistema, consta de cuatro etapas fundamentales:

1. Adquisición de series temporales.
2. Extracción de características.
3. Selección de características.
4. Construcción del modelo de clasificación de estados.

La metodología propuesta se encuentra representada en la Figura 3.10 y cada uno de sus componentes serán explicados a continuación.

3.4.1. Adquisición de series temporales

La adquisición de las series temporales, en la mayoría de casos, se lleva a cabo mediante un sistema de adquisición de datos compuesto por dispositivos específicos para la obtención de las mediciones de la variable predictora. Más detalles de un sistema de adquisición de datos y los elementos que lo componen se encuentra en la Sección 2.5 y un caso concreto puede ser visto en la Sección 2.5.1. En esta etapa se obtiene un conjunto de series de tiempo de una variable aleatoria, que se etiquetan con el estado al cual corresponden. La longitud de la serie temporal depende del proceso específico, sin embargo debe ser elegida de tal forma que contenga los eventos que distingan a un estado de los demás.

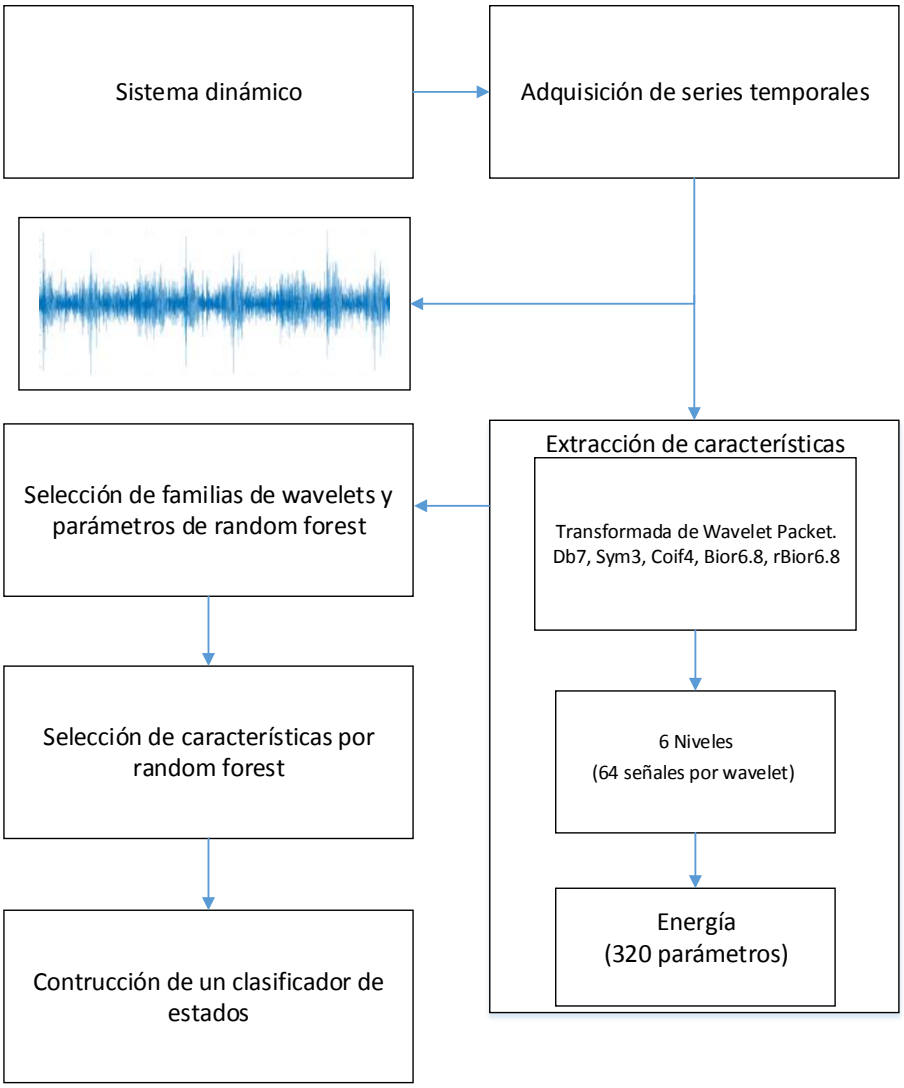


Figura 3.10: Metodología propuesta (enfoque clásico).

3.4.2. Extracción de características

Una serie de tiempo obtenida de una variable predictora que es accesible para una medición normalmente no muestra un comportamiento apreciable que permita relacionarla de forma directa con cambios en el estado de una variable no accesible pero de interés en un sistema dinámico. Dependiendo del caso concreto, está relación se puede mostrar más clara en una representación diferente de la serie. Esta representación es elegida acorde a las características intrínsecas de la variable medible y en todos los casos es dependiente de la aplicación.

Es por está razón que la etapa de extracción de características en el enfoque clásico de aprendizaje automático es considerada la más importante y a la vez la que demanda más tiempo. Por ejemplo en el trabajo [99], para el diagnóstico de fallos en maquinaria reciprocante se propone el uso de estadísticos clásicos de las series temporales (el valor medio, *rms* y el factor de forma) en conjunción con una técnica basada en la Teoría de la Información para la extracción de una característica adicional desde la serie temporal de la señal de vibración. Con el mismo fin, han sido propuestas otras técnicas más avanzadas para la extracción de características, como la mostrada en el trabajo [107], donde se propone el uso de la descomposición modal empírica de la serie de tiempo para posteriormente realizar el cómputo de su energía. Como resultado, se muestra que la energía en algunas funciones de modo intrínseco cambian con la presencia de un fallo en un rodamiento y se propone el uso de la entropía como mecanismo de cuantificación del cambio.

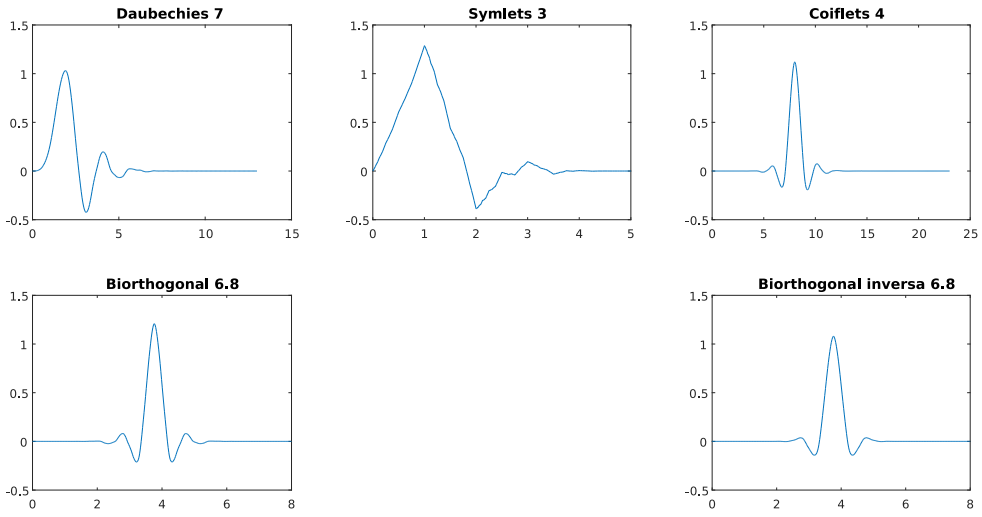


Figura 3.11: Wavelets madre.

En este capítulo proponemos el uso de Wavelet Packet Decomposition (WPD), un método que permite obtener información en el dominio del tiempo y la frecuencia. Más detalles acerca de WPD y en general del análisis tiempo-frecuencia pueden

encontrarse en la Sección 2.4. La representación en tiempo-frecuencia permite obtener la información conjunta de frecuencias específicas ocurriendo en un tiempo determinado.

Tal y como se muestra en la fase extracción de características de la Figura 3.10, se descompone la serie de tiempo de entrada mediante una wavelet específica en 6 niveles, donde en el sexto se obtienen 64 sub-series temporales denominadas *coeficientes*. Posteriormente, se calcula la energía de cada coeficiente, generando así un vector de 64 valores reales. Este proceso es aplicado de forma similar para 5 tipos de wavelets madres diferentes: daubechies 7 (db7), symlet 3 (sym3), coiflet 4 (coif4), biorthogonal $6{,}8$ (bior6.8) y biorthogonal inversa $6{,}8$ (rbior6.8) (ver Figura 3.11). Todos los vectores obtenidos del proceso anterior se concatenan, obteniendo finalmente un vector de características de 320 elementos. La extracción de características se aplica a todos los elementos del conjunto de series de tiempo representativas de los estados de la variable aleatoria que se obtuvo en la fase de adquisición de datos, obteniendo finalmente un conjunto de datos uniforme sobre el que trabajar.

3.4.3. Selección de familias de Wavelet y parámetros de Random Forest

En esta etapa, comenzamos dividiendo el conjunto de datos obtenido en la etapa de extracción de características en dos subconjuntos: entrenamiento y prueba. Como se verá más adelante, con la metodología propuesta no se requiere la presencia de un subconjunto de validación[1], obviamente sin riesgo alguno a sesgar los resultados.

Para encontrar el mejor subconjunto de familias de wavelets junto con los parámetros adecuados para el modelo se propone una búsqueda voraz. Concretamente, se crea un modelo de Random Forest con: cada posible subconjunto de familias de wavelets, cada posible número de árboles en un rango de $nMinArboles$ a $nMaxArboles$, y número de variables aleatorias desde 1 a $familias - en - subconjunto \times 64$. Estos modelos son evaluados por medio del oob-error de forma independiente.

Finalmente, se elige el subconjunto de familias, número de árboles y número de variables aleatorias que dan, en conjunto, el menor oob-error. Si existen varios subconjuntos de familias que con un valor adecuado de parámetros dan el mismo oob-error, entonces se elige el subconjunto con menor número de familias. Si además el número de familias es el mismo en los subconjuntos con menor oob-error se elige el subconjunto que, sumados el número de árboles y características aleatorias, da el menor valor.

El criterio de selección anteriormente planteado se enfoca en elegir el conjunto de familias y parámetros que represente la menor carga computacional para el modelo final.

[1]Para metodologías clásicas con redes neuronales es aconsejable contar con un subconjunto adicional de validación para evitar el sobreajuste del modelo o sesgarlo al conjunto de prueba.

3.4.4. Selección de características con Random Forest

De la etapa anterior finalmente se obtiene un subconjunto reducido de características que pertenecen solamente al subconjunto de familias de wavelet que mejor desempeño tienen para la tarea específica. Sin embargo, que una familia de wavelets sea, en general, apropiada para la aplicación no significa que todas sus componentes (energía de sus coeficientes) también lo sean. Para explicar esto es necesario ahondar en la interpretación física de un coeficiente.

Un coeficiente del nivel inferior en la descomposición por wavelet packet representa la serie de tiempo original en un intervalo de frecuencia específica. Esto significa que éste puede verse como el resultado de un proceso de filtrado que ha sido ajustado por las características de una wavelet madre específica. Es necesario resaltar que para una variable aleatoria de interés, en un sistema dinámico específico, no todos los intervalos de frecuencia son informativos para la identificación de los estados, por lo que se propone la aplicación de un procedimiento de ranking de características guiado, en primera instancia, por el propio modelo de Random Forest que es presentado en el trabajo original de Breiman [10], y que resumimos a continuación:

1. Obtener el oob-error inicial.

2. Para todos los conjuntos de datos de instancias fuera de las consideradas en cada árbol (los que se conoce como *fuera de la mochila*, el mismo procedimiento que se utiliza para el cálculo del oob-error), seleccionar la característica que se desea evaluar.

3. Permutar de forma aleatoria la característica seleccionada.

4. Evaluar los árboles con los conjuntos fuera de la mochila modificados.

5. Obtener el nuevo oob-error.

6. Calcular la diferencia entre el oob-error inicial y el nuevo oob-error como una representación de su importancia.

7. Repetir los pasos anteriores para todas las características que se deseen evaluar.

En nuestra propuesta, el procedimiento anterior es aplicado a todas las características del conjunto de datos de entrenamiento, logrando obtener de esta manera un vector con los valores de importancia de las características.

Posteriormente, se seleccionan solo aquellas que sobrepasan un umbral, que se elige de acuerdo a la aplicación específica. Sin embargo, es necesario considerar que las características que sobrepasen el umbral seleccionado debe ser mayor o igual que el valor para el parámetro del número de características aleatorias.

3.4.5. Construcción de un clasificador de estados

Para la construcción de un clasificador de estados, del conjunto de entrenamiento se eligen solamente las características con mayor importancia siguiendo el criterio detallado anteriormente. Con este nuevo conjunto se genera un modelo de Random Forest para una tarea de clasificación. Por otro lado, y al igual que en el conjunto de entrenamiento, en el conjunto de prueba solamente se seleccionan las características con mayor importancia (acorde a lo obtenido con el conjunto de entrenamiento). Posteriormente, el modelo final es evaluado en el conjunto de prueba utilizando las métricas clásicas para clasificación que fueron presentadas en el Apartado 2.7.4.

3.5. Evaluación en diagnóstico de fallos

Los sistemas de transmisión de potencia mecánica son piezas importantes en diferentes tipos de máquinas para aplicaciones industriales, siendo las cajas de engranes rectos las más utilizados para aplicaciones de baja y media potencia. Los estudios realizados en el área del mantenimiento basado en la condición han permitido en varios casos determinar en cada instante el estado real de una máquina proporcionando herramientas importantes para prevenir y/o detectar fallos tempranos en sistemas mecánicos. Para este proceso se utilizan técnicas no destructivas, siendo el análisis de vibraciones la más utilizada, con el fin de realizar acciones de mantenimiento preventivo que eviten costos futuros por daños graves o cortes inesperados del proceso.

Las técnicas del mantenimiento basado en la condición tuvieron sus inicios hace mucho tiempo atrás, por lo que existe una gran variedad de ellas. Un primer enfoque básico son las técnicas para el análisis de desechos de aceite. También existen técnicas avanzadas como: el análisis de movimiento angular, análisis de vibraciones, análisis basado en modelos, y modelado matemático. Una revisión detallada se puede encontrar en [44].

Por otra parte, el modelo de Random Forest ha sido aplicado en múltiples áreas de la ingeniería. Por ejemplo en [78] se utiliza para la clasificación de siete diferentes cubiertas terrestres, donde además se busca comparar su rendimiento con el de Máquinas de Soporte Vectorial en términos de exactitud de clasificación y tiempo de entrenamiento. Concretamente, el experimento busca clasificar las siguientes cubiertas: trigo, patata, remolacha azucarera, cebolla, guisantes, lechuga y frijoles a partir de imágenes por satélite. Los datos fueron adquiridos del Mapeador Temático Mejorado Landsat-7, donde cada imagen pertenece a las bandas espectrales 1-5 y 7, en una área agrícola de Littleport, Cambridgeshire, UK, en las temporadas de cultivos. El conjunto de datos consta de un total de 4737 imágenes para las siete clases, y cada imagen cuenta con un área de 307 pixels (columnas) por 330 pixels (filas). Se implementó un muestreo aleatorio estratificado, con 2700 muestras que fueron destinadas al conjunto de entrenamiento y 2037 muestras al conjunto de prueba. Los parámetros para Random Forest fueron de 3 características en cada nodo y un total de 100

árboles, consiguiendo una exactitud del 88.37 % en 12.98 seg a diferencia del 87.9 % obtenido con SVM en 18 seg.

En [65] se implementa un RF para la detección de fallas en turbinas de gas. El documento argumenta los altos costos y las consecuencias catastróficas que pueden causar las fallas en las aspas de las turbinas, cuyo análisis es complicado por el alto nivel de ruido que se presenta en todas las mediciones. Los resultados de RF fueron comparados con otros clasificadores como redes neuronales, arboles de decisión, Naive Bayes y KNN. Se utilizaron 12 instrumentos de medición diferentes cuyas mediciones fueron tomadas por cada posible combinación entre las 5 condiciones operacionales del motor: la primera con el motor en estado saludable, y las otras cuatro en condición de fallas (suciedad del motor, contaminación individual de las aspas del rotor, aspa torcida del rotor individual y reposicionamiento del aspa del rotor). También se configuraron cuatro cargas diferentes del motor (carga completa, media carga, cuarto de carga y sin carga), y finalmente se realizaron mediciones para 2 frecuencias diferentes de muestreo, obteniendo 864 instancias. Para cada instrumento, todas y cada una de las mediciones anteriores consisten de 27 características formados por la diferencia espectral de los primeros 27 armónicos de la frecuencia de rotación del eje del rotor. Debido el gran número de sensores y características se implementaron métodos de selección de características e identificación de valores atípicos. Para el primero se utilizó el indice Gini para hacer un ranking de las características de forma descendente, y, para eliminar el ruido en las características se implementaron técnicas estadísticas. Finalmente, se implementaron dos versiones del clasificador RF: Bosques de Entrada Aleatoria (RE), y Bosques Combinados al Azar (RC). Las métricas a evaluar fueron *precision* y *recall*. Los mejores resultados se obtuvieron al implementar 500 árboles y un total de 6 características aleatorias, teniendo como ganador a RC con una ligera diferencia. De forma general RC y RE son mas precisos que redes neuronales, árboles de decisión, Naive Bayes y KNN.

En [79] se muestra el uso de RF para la selección de características y detección de fallas mecánicas en un motor de inducción. En primer lugar, se capturan las señales de vibración producida por la máquina (debido a que la vibración es uno de los parámetros vitales, además de ser muy usado en la detección de fallas, del motor de inducción). La frecuencia de muestreo fue de 12KHz para 4 estados (clases) de la máquina: normal, falla de la pista de rodadura, fallo de bola en el rodamiento, y fallo de la pista externa. Se tomaron 160 instancias para el estado normal y 480 para cada uno de los estados restantes. La longitud de cada señal capturada fue de 0.25seg. Posteriormente, se extrajeron 15 características estadísticas de cada señal. Para la etapa de clasificación implementaron dos clasificadores: Redes Neuronales Artificiales (RNA), con pesos inicializados aleatoriamente desde una distribución normal estándar, y Random Forest, configurado con 1000 árboles. Como resultado se obtuvo RF como mejor clasificador, con 2 falsos positivos, frente a los 14 de RNA, teniendo como referencia un total de 1600 instancias. Finalmente se utilizó el diagrama de importancias entregado por RF para realizar una selección de atributos, siendo los más importantes el Factor de forma, la media cuadrática, el valor pico, la

varianza, el rms y la entropía.

En este trabajo, para evaluar la metodología anterior, se propone su aplicación en dos tareas de mantenimiento basado en la condición: el diagnóstico de fallos en rodamientos, y el diagnóstico de fallos en engranes rectos, que detallamos a continuación.

3.5.1. Diagnóstico multi-fallos en rodamientos

La aplicación de la metodología para el diagnóstico de fallos en rodamientos se realiza sobre las señales obtenidas en la configuración experimental detallada en la Subsección 2.6.4. Para esta aplicación, la variable de interés representa el estado físico de los elementos del rodamiento (pista interna, pista externa y elemento rodante) teniendo de esta manera 7 estados que se desean clasificar (ver Tabla 2.6). Los resultados obtenidos en cada fase de la metodología para esta tarea concreta son:

Extracción de características

En esta fase se descompuso cada señal de las disponibles mediante las 5 familias de wavelets para el posterior cómputo de la energía en sus coeficientes. Como resultado se obtuvo un conjunto de datos de 315 instancias con 320 características.

Selección de familias de wavelet y parámetros de Random Forest

En la etapa de selección de parámetros y mejor subconjunto de familias de wavelet se configuró $nMinArboles = 15$ y $nMaxArboles = 1000$. Los mejores resultados para cada combinación de familias de wavelet se muestran en la Tabla 3.1, donde se puede observar que los subconjuntos de familias de wavelet {db7, coif4, bior6.8}, {coif4, bior6.8, rbior6.8}, {db7, sym3, coif4, bior6.8}, {db7, sym3, bior6.8, rbior6.8}, {db7, coif4, bior6.8, rbior6.8} y {db7, sym3, coif4, bior6.8, rbior6.8} poseen el mismo valor inferior ($3{,}64 \times 10^{-}2$) de oob-error. Al elegir el de menor tamaño se obtienen dos subconjuntos en conflicto: {db7, coif4, bior6.8} y {coif4, bior6.8, rbior6.8}. Sin embargo, la suma de sus parámetros da 107 y 72 respectivamente, lo que permite elegir al subconjunto {coif4, bior6.8, rbior6.8} con 16 características aleatorias y 56 árboles como la mejor combinación de familias y parámetros con el menor coste computacional para la tarea. Tras este paso el número de características para la tarea queda reducido a 192.

Selección de características

Dado el conjunto de familias y parámetros obtenidos en el paso anterior se procede con la selección de las características más relevantes para la tarea, determinando la importancia de cada característica.

Wavelets	Características	Árboles	oob-error×100
db7	14	58	4.55
sym3	8	561	5.00
coif4	39	69	4.09
bior6.8	40	20	4.09
rbior6.8	14	74	4.09
db7, sym3	64	41	4.55
db7, coif4	44	52	4.55
db7, bior6.8	24	43	4.09
db7, rbior6.8	13	403	5.00
sym3, coif4	21	35	4.09
sym3, bior6.8	67	9	4.09
sym3, rbior6.8	14	598	4.55
coif4, bior6.8	16	241	4.09
coif4, rbior6.8	16	255	4.09
bior6.8, rbior6.8	10	40	4.55
db7, sym3, coif4	48	22	4.09
db7, sym3, bior6.8	61	64	4.09
db7, sym3, rbior6.8	10	984	4.55
db7, coif4, bior6.8	25	82	3.64
db7, coif4, rbior6.8	12	405	4.55
db7, bior6.8, rbior6.8	16	62	4.09
sym3, coif4, bior6.8	25	53	4.09
sym3, coif4, rbior6.8	66	168	4.09
sym3, bior6.8, rbior6.8	16	25	4.09
coif4, bior6.8, rbior6.8	16	56	3.64
db7, sym3, coif4, bior6.8	125	62	3.64
db7, sym3, coif4, rbior6.8	63	173	4.09
db7, sym3, bior6.8, rbior6.8	56	34	3.64
db7, coif4, bior6.8, rbior6.8	93	33	3.64
sym3, coif4, bior6.8, rbior6.8	16	62	4.09
db7, sym3, coif4, bior6.8, rbior6.8	60	20	3.64

Cuadro 3.1: Mejores parámetros por cada subconjunto de familias de wavelet.

El resultado de este proceso se muestra en la Figura 3.12, donde se aprecia que las características 2, 66 y 130 (que corresponden a la energía de los coeficientes 2 de coif4, 2 de boir6.8 y 2 de rbior6.8) sobresalen con un valor de importancia por encima de 0.04. Además, las siguientes características sobresalientes son la 1, 65 y 129 (correspondientes a la energía de los primeros coeficientes de las tres familias seleccionadas). Este resultado indica que para esta tarea, las componentes 1 y 2 de baja frecuencia de cada wavelet aportan la mayor cantidad de información discriminante.

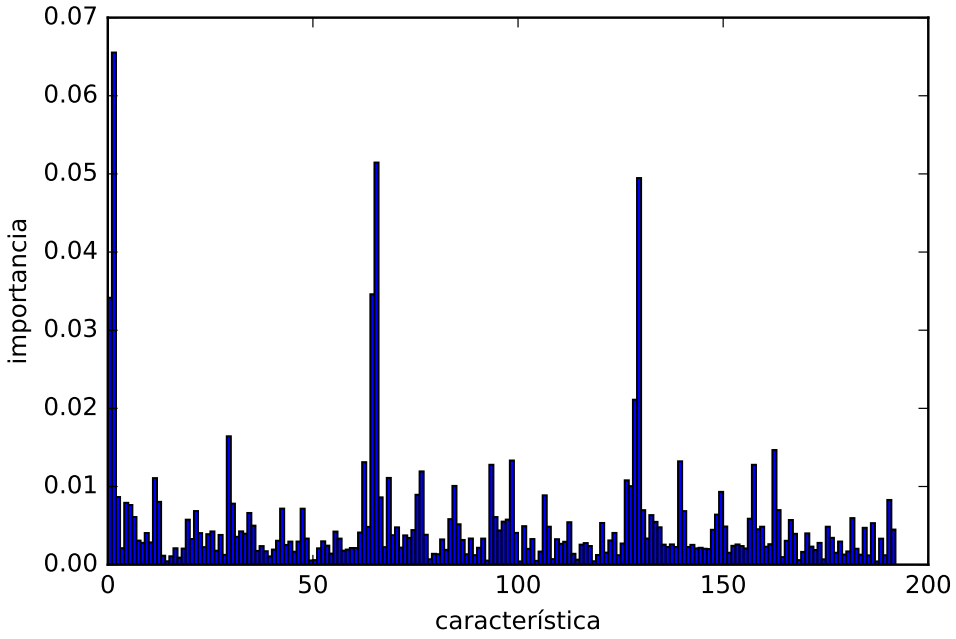


Figura 3.12: Ranking de características para diagnóstico de multi-fallos en rodamientos.

Para la selección del umbral se realiza una búsqueda manual desde 0 a 0.008[2] con incrementos de 0.001. Con cada valor de umbral se seleccionan aquellas características que lo sobrepasan, y posteriormente se generan modelos de clasificación por Random Forest con cada conjunto de características y se evalúa su oob-error, dando los resultados mostrados en la Tabla 3.2.

El mejor valor de umbral obtenido es 0.003 con un total de 101 características seleccionadas.

[2]Este es el valor máximo con el que se obtiene un número de características mayor o igual al parámetro de número de características aleatorias.

Umbral	Núm. de carac.	oob-error$\times 100$
0.000	192	7.27
0.001	177	6.36
0.002	145	6.36
0.003	101	5.46
0.004	76	6.82
0.005	52	6.82
0.006	40	5.91
0.007	31	5.91
0.008	26	6.36

Cuadro 3.2: Resultados de número de características seleccionadas y oob-error por cada valor de umbral.

Construcción de un clasificador de estados

Finalmente, con las familias (coif4, bior6.8, rbior6.8), parámetros ($nArboles = 82$, $nCaracteristicasAleatorias = 25$) y características seleccionadas (101 características) se construye el modelo de clasificación de estados.

Este modelo se evalúa con el conjunto de prueba obteniendo, para cada estado, las métricas mostradas en la Tabla 3.3, donde se puede observar una disminución de la precisión en el estado P1, P3 y P5, lo que indica que el modelo clasificó instancias correspondientes de otros estados dentro de los mencionados. De igual forma, es apreciable la disminución de recall para los estados P6 y P7, lo cual indica que instancias perteneciente a estos estados fueron clasificados en otros. De lo anterior se puede deducir que esos otros estados son P1, P3 y P5. Sin embargo, con la información suministrada en esa tabla no es posible determinar específicamente con qué estados se produce la confusión. La métrica F-score da una información general por clase equilibrando la precisión y el recall.

Para un análisis detallado de los errores cometidos por el modelo se presenta la matriz de confusión en la Tabla 3.4, donde se observa que el mayor número de errores del clasificador se da para instancias del estado 7 (correspondiente a fallo en pista externa en el rodamiento 1 y elemento rodante en el rodamiento 2), que son confundidas con el estado 3 (correspondiente a fallo en pista externa en el rodamiento 1 y condición normal en el rodamiento 2), y con el estado 1 (que corresponde a condición normal en los 2 rodamientos). Algo similar sucede con el error cometido por el clasificador para instancias del estado 6 (que corresponde a fallo en pista interna en el rodamiento 1 y fallo en elemento rodante en el rodamiento 2), que son confundidas con el estado 5 (correspondiente a fallo en pista interna del rodamiento 1 y fallo en pista externa del rodamiento 2).

Estado	Precisión	Recall	F-score
P1	0.88	1.00	0.93
P2	1.00	1.00	1.00
P3	0.78	1.00	0.88
P4	1.00	1.00	1.00
P5	0.87	1.00	0.93
P6	1.00	0.86	0.92
P7	1.00	0.57	0.73

Cuadro 3.3: Métricas para el modelo de clasificación de estados evaluado con el conjunto de prueba.

Estado	P1-P	P2-P	P3-P	P4-P	P5-P	P6-P	P7-P
P1-R	14	0	0	0	0	0	0
P2-R	0	13	0	0	0	0	0
P3-R	0	0	14	0	0	0	0
P4-R	0	0	0	13	0	0	0
P5-R	0	0	0	0	13	0	0
P6-R	0	0	0	0	2	12	0
P7-R	2	0	4	0	0	0	8

Cuadro 3.4: Matriz de confusión.

Lo anterior indica que para el rodamiento 1, que es el lugar en donde se adquirió la señal de vibración, en realidad solamente se cometieron 2 errores con el estado normal. El resto de errores se deben a la incapacidad del modelo para estimar los fallos existentes en el rodamiento 2. De donde podemos concluir que la señal de vibración capturada en el rodamiento 1 no contiene información suficiente para estimar el estado del rodamiento 2.

3.5.2. Diagnóstico de fallos en engranes rectos

La metodología propuesta ha sido aplicada también al diagnóstico de fallos en engranajes rectos con las señales obtenidas en la configuración experimental detallada en la Subsección 2.6.1.

Para esta aplicación, la variable de interés representa el estado físico de distintos elementos de la caja de engranes (piñón, engrane y eje) teniendo de esta manera 10 estados que se desean clasificar (ver Tabla 2.1). A continuación mostramos los

resultados obtenidos en cada fase de la metodología propuesta:

Extracción de características

En esta fase se obtiene el árbol de descomposición de cada señal de las 1500 disponibles mediante las 5 familias de wavelets para el posterior cómputo de la energía en sus coeficientes. Como resultado se obtuvo un conjunto de datos de 1500 instancias con 320 características.

Selección de familias de wavelet y parámetros de Random Forest

En la etapa de selección de parámetros y mejor subconjunto de familias de wavelet se configuró $nMinArboles = 15$ y $nMaxArboles = 1000$. Los mejores resultados para cada combinación de familias de wavelet se muestran en la Tabla 3.5, donde se observa que el subconjunto de wavelets compuesto por {db7, sym3, bior6.8} tiene el menor oob-error. Al contrario que lo sucedido en el ejemplo de rodamientos, aquí no es necesaria la aplicación de las reglas adicionales para la selección del mejor subconjunto de wavelets y parámetros del modelo, por lo que se elige el subconjunto mencionado con los parámetros de 13 características aleatorias y 968 árboles como las mejores parámetros para la tarea. Tras este paso el número de características queda reducido a 192.

Selección de características

De forma similar al ejemplo anterior, dado el conjunto de familias y parámetros obtenidos en el paso anterior se procede con la selección de las características más relevantes para la tarea, para lo que se determina la importancia de cada característica. El resultado de este proceso se muestra en la Figura 3.13, donde se puede apreciar que la energía de los coeficientes 4 y 7 de cada familia son los que mayor importancia tienen. Este resultado indica que, para esta tarea, las componentes frecuenciales 4 y 7 de cada wavelet aportan la mayor cantidad de información discriminante.

Para la selección del umbral se realiza una búsqueda manual desde 0 a 0.011[3] con incrementos de 0.001. Con cada valor de umbral se seleccionan aquellas características que lo sobrepasan, y luego se generan modelos de clasificación por Random Forest con cada conjunto de características y se evalúa su oob-error, dando los resultados mostrados en la Tabla 3.6. El mejor valor de umbral obtenido es 0.003 con un total de 126 características seleccionadas.

[3]Este es el valor máximo con el que se obtiene un número de características mayor o igual al parámetro de número de características aleatorias.

Wavelets	Características	Árboles	oob-error×100
db7	16	813	5.33
sym3	7	460	5.62
coif4	16	390	9.91
bior6.8	17	660	7.14
rbior6.8	11	737	7.33
db7, sym3	6	885	4.48
db7, coif4	13	557	5.33
db7, bior6.8	16	896	4.38
db7, rbior6.8	19	848	4.48
sym3, coif4	10	647	5.52
sym3, bior6.8	8	583	4.48
sym3, rbior6.8	8	313	4.57
coif4, bior6.8	30	635	6.29
coif4, rbior6.8	30	638	6.48
bior6.8, rbior6.8	15	960	7.05
db7, sym3, coif4	12	902	4.57
db7, sym3, bior6.8	13	968	3.71
db7, sym3, rbior6.8	11	933	3.91
db7, coif4, bior6.8	8	658	4.67
db7, coif4, rbior6.8	18	834	4.57
db7, bior6.8, rbior6.8	12	675	4.95
sym3, coif4, bior6.8	25	710	4.57
sym3, coif4, rbior6.8	24	973	4.67
sym3, bior6.8, rbior6.8	12	608	4.38
coif4, bior6.8, rbior6.8	15	958	6.57
db7, sym3, coif4, bior6.8	27	802	3.91
db7, sym3, coif4, rbior6.8	13	813	3.91
db7, sym3, bior6.8, rbior6.8	10	936	4.00
db7, coif4, bior6.8, rbior6.8	12	646	4.76
sym3, coif4, bior6.8, rbior6.8	21	737	4.67
db7, sym3, coif4, bior6.8, rbior6.8	12	977	4.1

Cuadro 3.5: Mejores parámetros por cada subconjunto de familias de wavelet.

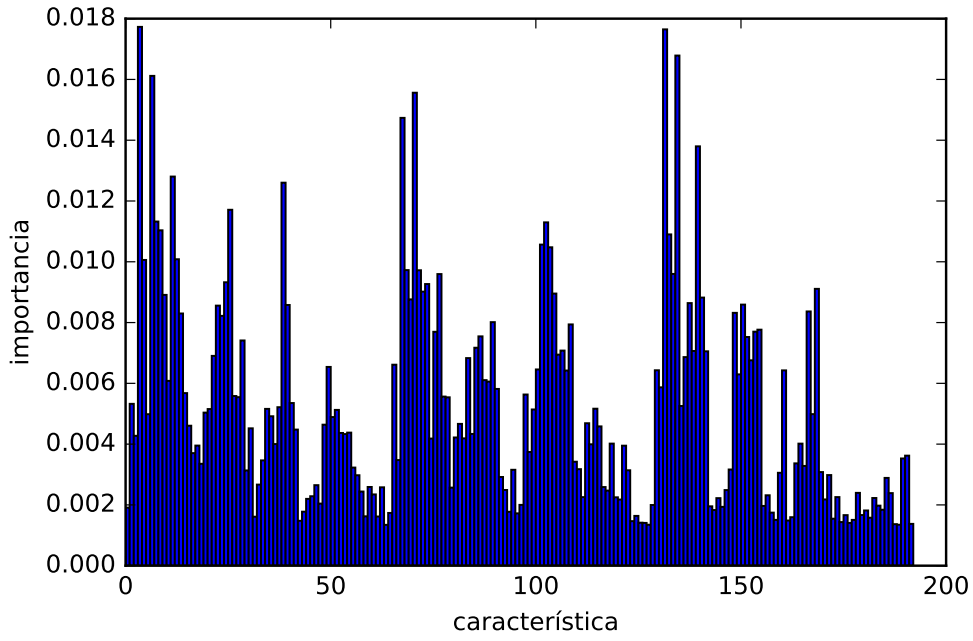


Figura 3.13: Ranking de características para diagnóstico de fallos en engranajes rectos.

Umbral	Núm. de carac.	oob-error$\times$100
0.000	192	4.19
0.001	192	4.19
0.002	155	4.48
0.003	126	3.62
0.004	105	4.19
0.005	83	3.81
0.006	65	4.38
0.007	50	5.91
0.008	39	8.00
0.009	26	11.24
0.010	18	13.33
0.008	13	18.67

Cuadro 3.6: Resultados de número de características seleccionadas y oob-error por cada valor de umbral.

Estado	Precisión	Recall	F-score
P1	1.00	0.98	0.99
P2	0.78	0.84	0.81
P3	0.78	0.78	0.78
P4	0.93	0.89	0.91
P5	1.00	0.93	0.97
P6	0.98	1.00	0.99
P7	0.98	1.00	0.99
P8	0.98	1.00	0.99
P9	0.98	0.98	0.98
P10	0.95	0.93	0.94

Cuadro 3.7: Métricas para el modelo de clasificación de estados evaluado con el conjunto de prueba.

Construcción de un clasificador de estados

Finalmente, con las familias {db7, sym3, bior6.8}, parámetros ($nArboles = 968$, $nCaracteristicasAleatorias = 13$) y características seleccionadas (126 características) se construye el modelo de clasificación de estados.

Este modelo se evalúa con el conjunto de prueba obteniendo, para cada estado, las métricas mostradas en la Tabla 3.7, donde se puede observar una precisión menor que 0.9 en los estados P2 y P3, lo que indica que el modelo clasificó instancias correspondientes de otros estados dentro de los mencionados. De igual forma, es apreciable la disminución de recall para los estados P2, P3 y P4, lo cual indica que instancias perteneciente a estos estados fueron clasificados en otros. De lo anterior se puede deducir que esos otros estados son en su mayoría P2 y P3.

Para un análisis detallado de los errores cometidos por el modelo se presenta la matriz de confusión en la Tabla 3.8, donde se observa que el mayor número de errores del clasificador se da para instancias del estado 3 (correspondiente a fallo por ruptura del diente al $10\,\%$ en el engrane), que son confundidas con daño por fisura en el engrane (estado 2). La siguiente mayor confusión se da entre instancias de daño por fisura en el engrane (estado 2), que son incorrectamente clasificadas como fallo por ruptura del diente al $10\,\%$ en el engrane (estado 3), lo que hace evidente que las formas de vibración para estos dos estados son similares entre sí.

Estado	P1-P	P2-P	P3-P	P4-P	P5-P	P6-P	P7-P	P8-P	P9-P	P10-P
P1-R	44	0	0	0	0	0	1	0	0	0
P2-R	0	38	6	0	0	0	0	0	0	1
P3-R	0	9	35	0	0	0	0	0	0	1
P4-R	0	1	4	40	0	0	0	0	0	0
P5-R	0	0	0	2	42	1	0	0	0	0
P6-R	0	0	0	0	0	45	0	0	0	0
P7-R	0	0	0	0	0	0	45	0	0	0
P8-R	0	0	0	0	0	0	0	45	0	0
P9-R	0	0	0	1	0	0	0	0	44	0
P10-R	0	1	0	0	0	0	0	1	1	42

Cuadro 3.8: Matriz de confusión.

CAPÍTULO 4

APRENDIZAJE DE CARACTERÍSTICAS DE SERIES DE TIEMPO

4.1. Introducción

En el capítulo anterior pudimos observar el uso de técnicas de aprendizaje automático para abordar la identificación de estados en un sistema dinámico y su aplicación a tareas de diagnóstico y estimación de la severidad del daño en elementos de una maquinaria rotativa. La resolución de estas tareas haciendo uso de dichas técnicas fue posible, en gran medida, gracias al esfuerzo puesto en el proceso de extracción de características. Como se puso de manifiesto en dicho capítulo, la elección de los mejores extractores de características (wavelets), desde las cuales se construye el clasificador, se realizó de forma manual a partir de un amplio conocimiento acerca del sistema dinámico y las leyes físicas que lo gobiernan.

En general, el proceso de extracción de características puede ser tan diverso como lo requiera el dominio de la aplicación. Incluso dentro de un dominio como en el que hemos enfocado nuestro estudio, el campo del análisis de señales de vibración, podemos encontrar ejemplos en los que este proceso puede precisar solamente del cálculo de métricas estadísticas en los distintos dominios de representación de la señal, como muestra en [94] para el diagnóstico de fallos en rodamientos bajo condiciones de operación estacionarias, y otros ejemplos en los que se puede requerir un análisis más profundo, y en principio menos automatizable, de la señal, como el realizado por [59], donde se propone una descomposición de la señal de vibración para identificar una componente que permita diferenciar fallos en cajas de engranes. La aplicación de las diferentes técnicas depende, entre otros factores, de las condiciones de operación de velocidad y carga en las que fueron adquiridas las señales de vibración, así como de los elementos mecánicos que conforman la máquina de

la que se registran las vibraciones. Por todo ello, resulta complicado encontrar una metodología general que, a priori, se pueda aplicar directamente para la evaluación del daño.

Como hemos visto, cuando las variables mencionadas son conocidas parcial o totalmente, es posible el diseño de métodos basados en el procesamiento de señales para encontrar patrones que permitan resolver la tarea concreta. Pero, ¿qué sucede cuando se dispone de un conocimiento limitado acerca de la naturaleza del proceso subyacente?, ¿se podría, en cierta medida, prescindir del conocimiento experto humano?

Para responder estas preguntas nos enfocamos en un objetivo inicial que sitúa, en primer lugar, el proceso de automatizar la extracción de características de las señales de vibración o cualquier representación informativa relacionada, por ejemplo, su espectro (dominio de la frecuencia) o su espectrograma (dominio tiempo-frecuencia). Sin embargo, es conveniente notar las dificultades intrínsecas que conlleva el diseño de un proceso para la extracción automática de características. En primer lugar, las series temporales o espacio-temporales obtenidas de la medición en la maquinaria bajo condiciones de operación no-estacionarias presentan cambios tanto en posición como en forma en cualquiera de los dominios que se pueda representar[59]. En segundo lugar, el espacio dimensional a explorar en la búsqueda de patrones usando las técnicas clásicas de aprendizaje automático es demasiado grande, por lo que resulta computacionalmente ineficiente (y, en la mayoría de los casos, imposible).

En consecuencia, es necesario encontrar un nuevo enfoque orientado a las ideas originales de aprendizaje, donde se fusionen áreas como el procesamiento de señales, la extracción y selección de características, la representación del conocimiento, el aprendizaje supervisado y no-supervisado, en un único modelo en el que cada área interactúe directamente con las demás y donde las estrategias de refinamiento y optimización se apliquen en conjunto.

En este capítulo se presenta una metodología que busca cumplir con algunos de los objetivos planteados. Para conseguir nuestros fines nos basaremos en el uso conjunto de Stacked Convolutional Autoencoders (SCAE) junto con Deep Convolutional Neural Networks (DCNN) como un método para la extracción jerárquica y no-supervisada de características, y mostraremos su uso en la evaluación de la severidad del daño en elementos mecánicos de una máquina rotativa, donde el proceso de extracción de características usa las series de tiempo bajo condiciones de operación estacionarias y no-estacionarias. La principal contribución de la propuesta que presentamos en este capítulo es la mejora en la exactitud de estimación en el nivel de severidad del daño. Para ello realizaremos una detección no-supervisada de una jerarquía de patrones en el dominio de tiempo-frecuencia, local o globalmente relacionados entre sí, usando una DCNN, que es mejorada mediante un SCAE usado para capturar patrones a priori, y para pre-inicializar los parámetros de la DCNN.

El resto de este capítulo está organizado como sigue. La Sección 4.2 presenta los fundamentos de las redes neuronales clásicas, las redes neuronales convolucionales, los autoencoders y SCAE. La Sección 4.3 describe en detalle el método propuesto

y su aplicación a la evaluación de la severidad del daño en elementos mecánicos a partir de señales de vibración. La Sección 4.4 muestra la experimentación realizada para validar el método en un caso concreto de severidad de daño en una caja de engranes helicoidales. Terminamos esta última sección mostrando una comparativa con diferentes métodos clásicos de extracción de características supervisados [94], y no-supervisados [71, 108].

4.2. Fundamentos teóricos

Para utilizar técnicas de aprendizaje automático con series temporales o espacio-temporales es necesario comenzar enfocando nuestra atención al comportamiento de esas técnicas como un sistema de procesamiento de señales, que en su mayoría se encuentran gobernados por procesos que realizan una descomposición de la señal entrante usando filtros que permiten extraer la información de interés. La información resultante de estos procesos se encuentra en forma de una nueva señal que contiene un rango de frecuencia específico determinado por las características del filtro. Habitualmente, los filtros que se usan son diseñados por expertos en la tarea específica y variarán de aplicación en aplicación.

En consecuencia, las técnicas de aprendizaje automático que deseamos obtener deben ser capaces de imitar el comportamiento de un sistema para el procesamiento de señales, lo que quiere decir que deben ser capaces de comportarse como filtros que extraen información acorde a ciertos criterios impuestos por la tarea específica. La diferencia fundamental que vamos a encontrar con respecto a un sistema de procesamiento de señales clásico es que el conocimiento necesario para la creación de los filtros no proviene de un experto en el dominio de aplicación, sino que debe extraerse de forma automática a partir de un conjunto de datos por medio de un proceso de aprendizaje.

Una herramienta de conexión existente entre el procesamiento de señales y las técnicas disponibles de aprendizaje automático la encontramos en el operador de convolución, situada en el núcleo de las operaciones habituales de diseño de filtros y también en modelos derivados de las redes neuronales, que se conocen de forma general como Redes Neuronales Convolucionales (CNNs), y que constituyen el modelo base sobre el que construiremos nuestra propuesta en este capítulo.

Por ello, comenzaremos dando unas notas acerca de este operador de convolución. A continuación presentaremos brevemente uno de los modelos de aprendizaje más conocidos, las Redes Neuronales Artificiales (ANN) junto con el algoritmo de entrenamiento basado en la retro-propagación del error, que es el fundamento de la mayoría de los algoritmos de entrenamiento usados en este tipo de redes. Posteriormente mostramos la Red Neuronal Convolucional (CNN) desde el punto de vista de un grupo de filtros que se aprenden a partir de un conjunto de señales, así como la modificación que es necesario realizar al algoritmo de retro-propagación para poder ser aplicado a este tipo de redes. Seguiremos presentando los Autoencoders (AE), un

tipo de red neuronal creada para el aprendizaje no-supervisado, a partir de la cual se construye una versión modificada del modelo convolucional, llamado Convolutional Autoencoder (CAE), y que consigue mejorar el desempeño de la CNN en la tarea que nos ocupa.

4.2.1. Operador de convolución

En general, la **convolución** es un operador que transforma dos funciones de entrada, f y g, en una función de salida que, de alguna forma, mide la superposición existente entre f y una trasladada de g, generalizando el concepto de media integral (que se consigue cuando g es la función característica de un intervalo).

En el caso de usar señales temporales como funciones, el operador de convolución, que denotaremos a partir de ahora por $*$, combina una señal de entrada $f(\cdot)$ con una señal patrón $g(\cdot)$, comúnmente llamada **kernel**. La señal resultante contiene la información destacada en base a algún criterio impuesto por las características del kernel. En una primera aproximación supondremos que la operación de convolución se hace respecto de una única variable, que desde el punto de vista de señales se interpreta como el tiempo.

Para funciones definidas en tiempo continuo, la operación de convolución viene dada por medio del operador integral:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau \tag{4.1}$$

La misma operación, en tiempo discreto, puede ser expresada como:

$$(f * g)[n] = \sum_{k=-\infty}^{\infty} f[k]\, g[n-k] \tag{4.2}$$

Este concepto puede ser generalizado para funciones definidas en dimesiones superiores. Por ejemplo, para un dominio bidimensional continuo la operación de convolución tendría la expresión:

$$(f * g)(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tau_1, \tau_2)\, g(x-\tau_1, y-\tau_2)\, d\tau_1\, d\tau_2 \tag{4.3}$$

y si el dominio fuera discreto la operación se podría expresar como:

$$(f * g)[x, y] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} f[k_1, k_2]\, g[x-k_1, y-k_2] \tag{4.4}$$

Las principales aplicaciones de esta operación se han dado en el campo del procesamiento de señales y hacen uso del **Teorema de Convolución**, que establece que

la transformada de Fourier de la convolución de dos funciones es igual al producto puntual de las transformadas de Fourier de cada función, o expresado formalmente:

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \odot \mathcal{F}\{g\} \tag{4.5}$$

La ecuación anterior muestra que el resultado de la convolución entre dos señales puede interpretarse como una versión filtrada de la señal f respecto a las componentes de frecuencia del kernel g. Lo que nos lleva a que, haciendo uso de un kernel adecuado, podemos resaltar los patrones deseados de cualquier señal. En este sentido, los kernel se convierten en el medio de destacar determinadas características de las señales, por lo que disponer de (o aprender) los kernel adecuados puede resolver nuestro objetivo de seleccionar las características adecuadas de las señales de vibración. Un ejemplo de convolución tanto en tiempo como en frecuencia se muestra en la Figura 4.1.

4.2.2. Red neuronal artificial

Una red neuronal artificial es un modelo de computación inspirado en las redes de neuronas existentes en el cerebro animal y que se cree que es el mecanismo responsable de la actividad inteligente. Su origen parte del trabajo realizado por McCulloch y Pitts en el artículo "*A logical calculus of the ideas immanent in nervous activity*"[70], que inspiró a Frank Rosenblatt para proponer el primer modelo de neurona artificial llamado perceptrón [85].

Perceptrón

El perceptrón [85] es el primer modelo computacional de neurona, y es el más sencillo de los modelos que se han definido y que intenta cubrir el funcionamiento básico de una neurona real: flujo entrante de señales, agregación, transformación, y flujo saliente de señales transformadas.

Este modelo está compuesto por un conjunto de entradas que reciben valores booleanos, $\{0, 1\}$, y una salida que es capaz de devolver algún valor booleano. De esta forma, intenta imitar de forma muy vaga un proceso binario de toma de decisiones basado en los estados e importancia de las entradas. Si las entradas son representadas mediante el vector $\mathbf{x} \in \{0, 1\}^d$, su peso asociado (o importancia) por $\mathbf{w} \in \mathbb{R}^d$, y su salida por $\hat{y}$, entonces el modelo queda totalmente definido por:

$$\hat{y} = \begin{cases} 0 & \text{si } \mathbf{w} \cdot \mathbf{x} \leq b \\ 1 & \text{si } \mathbf{w} \cdot \mathbf{x} > b \end{cases} \tag{4.6}$$

donde $b \in \mathbb{R}$ es un umbral de decisión. Típicamente, este modelo matemático de neurona es también representado por medio del modelo gráfico de la Figura 4.2.

El perceptrón puede cambiar su comportamiento (salida) ante un conjunto de entradas determinado, modificando sus pesos y el umbral, por lo que un proceso de

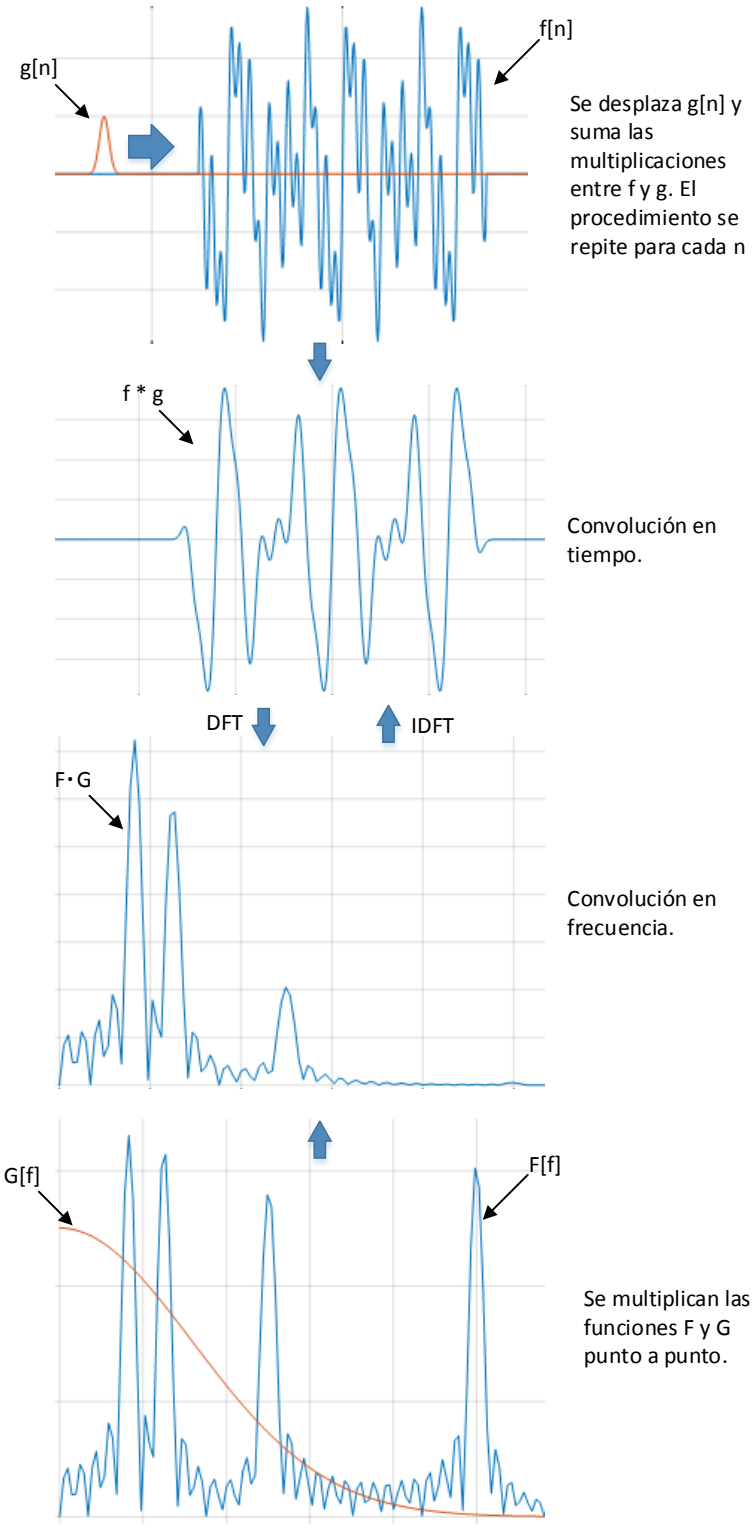


Figura 4.1: Procedimiento de la operación de convolución tanto en tiempo como en frecuencia y su relación por la transformada de Fourier directa e inversa.

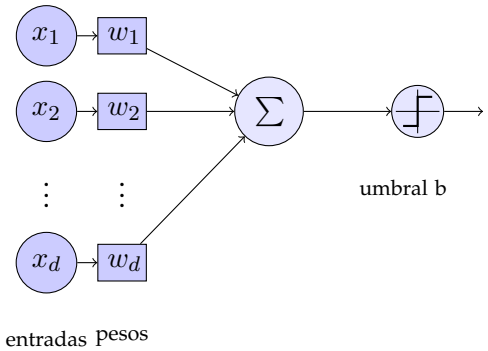


Figura 4.2: Modelo gráfico del perceptrón

aprendizaje para este modelo se reduce a encontrar los valores de pesos y de umbral adecuados para la tarea que se quiere resolver. Normalmente, se busca la mayor consistencia con la evidencia presente en un conjunto de datos de entrenamiento (es decir, menor error cometido), por lo que desde el punto de vista del aprendizaje se trataría de un modelo de aprendizaje supervisado.

En su concepción original, y con los valores de pesos y umbral adecuados, el modelo no es capaz de aproximar el funcionamiento de una puerta lógica NAND[1]. Este resultado muestra que el perceptrón es un modelo de cómputo con capacidades limitadas cercanas a simples operaciones de negación con la operación lógica "Y".

Neurona sigmoidal

Debido a las limitaciones que presenta el perceptrón simple como máquina de cómputo universal, se intentó abordar esta debilidad conectando múltiples perceptrones entre sí formando una red. Sin embargo, pese a que la capacidad computacional se incrementaba (aunque no lo suficiente), el proceso de aprendizaje también se hacía más complejo a medida que la red resultante crecía. Este fenómeno ocurre debido a la falta de control que existe ante una variación en un peso con respecto a la salida por la discontinuidad de la función escalón que se usa para el umbral, ya que, de acuerdo al modelo de computación de la ecuación (4.6) una mínima variación en un peso puede causar un cambio brusco (entre los valores 0 y 1) en la salida.

Una forma de eliminar esta falta de control es cambiando la función que actúa sobre el valor agregado de las señales de entrada por una función con propiedades analíticas más adecuadas (al menos, continuidad), y también permitiendo que los datos de entrada no sean binarios, sino números reales cualesquiera:

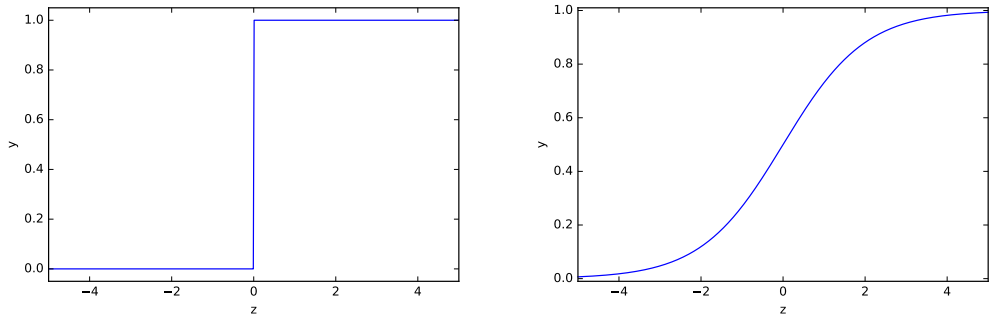
$$\hat{y} = \sigma(\mathbf{w} \cdot \mathbf{x} + b) \tag{4.7}$$

[1]La puerta NAND se considera la puerta universal debido que a partir de esta es posible construir cualquier otra compuerta y, por ende, cualquier circuito lógico.

donde $\mathbf{x} \in \mathbb{R}^n$ y σ es una función continua. En general, a esta función σ que actúa sobre el valor agregado de las señales (y umbral) se le denomina *función de activación* y, con el fin de facilitar la manipulación de este tipo de unidades durante el proceso de aprendizaje, suele considerarse la función no-lineal sigmoide siguiente como una de las funciones de activación más habituales:

$$\sigma(z) = \frac{1}{1 + \exp^{-z}} \tag{4.8}$$

La Figura 4.3 muestra la función de activación del perceptrón original (función escalón) y la función de activación para una neurona sigmoidal, donde se aprecia el cambio brusco que se produce en el estado de salida para el primer caso, frente el cambio progresivo y suavizado del segundo caso.



(a) Función de activación escalón

(b) Función de activación sigmoide

Figura 4.3: Funciones de activación para a) el perceptrón, y b) la neurona sigmoidal.

Red neuronal multicapa (MLP)

Una vez resuelto el problema de control sobre la neurona artificial individual, es posible agrupar neuronas sigmoidales e interconectarlas formando redes neuronales artificiales de mayor complejidad. Aunque como veremos no es la única configuración posible, la más utilizada a lo largo de los años anteriores ha sido la conocida como *red feedforward*, también conocida como *perceptrón multicapa* (MLP) [85], que está formada por capas de neuronas, donde cada neurona de una capa específica se conecta con todas las neuronas de la siguiente capa, tal y como muestra la Figura 4.4. El apelativo de feedforward proviene del hecho de que en esta arquitectura el flujo de datos solo se permite hacia capas inmediatamente posteriores, y no admite conexiones entre neuronas de una misma capa.

La red neuronal multicapa está compuesta por una primera capa donde se ingresa el vector de datos de entrada, llamada *capa de entrada*, una capa de salida con el resultado entregado por la red, que se llama *capa de salida*, y un conjunto de capas intermedias ordenadas jerárquicamente entre ambas capas, que se denominan

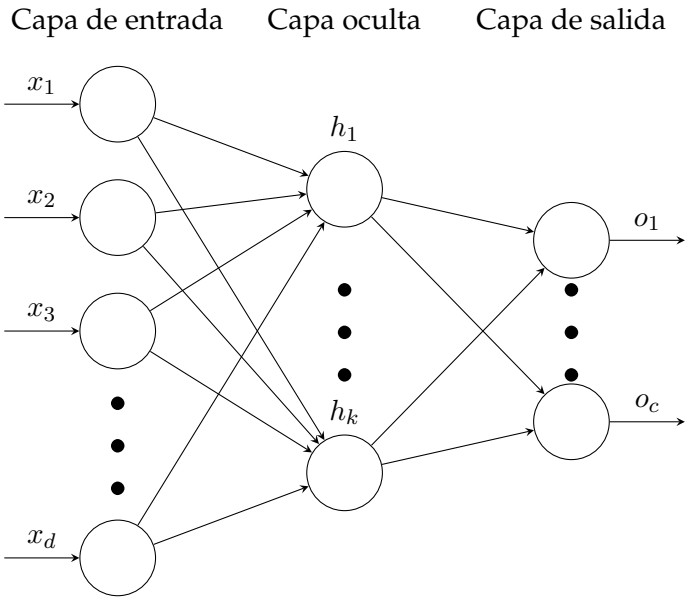


Figura 4.4: Red neuronal feedforward con una capa de entrada, una capa oculta, y una capa de salida.

globalmente *capas ocultas*, y que es donde se realizan proyecciones no lineales de la información que fluye a través de la red.

Aunque es evidente que una red que admita conexiones más flexibles puede aportar una mayor capacidad computacional, la razón por la que han sido las más utilizadas desde un punto de vista histórico se debe al hecho de que para ellas se dispone de métodos de entrenamiento eficientes, en los que la ausencia de ciclos en el flujo de información aseguran la convergencia de métodos tradicionales de optimización. A pesar de las limitaciones impuestas para que sean controlables desde el punto de vista del entrenamiento, este tipo de redes proporcionan un modelo de computación universal (véase el teorema de aproximación universal para ANN en [39]).

Aprendizaje Desde el punto de vista del aprendizaje supervisado, el aprendizaje de una red neuronal se obtiene por medio del ajuste de los pesos y bias para mejorar el desempeño de la red, que se entiende de forma habitual como la disminución del error existente entre la predicción de la red (o^L, la salida proporcionada por la última capa), y la salida deseada (y, disponible en el conjunto de datos sobre los que estamos haciendo aprendizaje). La función de error, también llamada función de coste, típicamente viene dada por el error medio cuadrático (MSE) del que ya hablamos anteriormente:

$$E = \frac{1}{2N}||y - o^L||_2^2 \tag{4.9}$$

donde N es el número de elementos que tiene el conjunto de datos. Aunque puede ser cualquier función derivable y decreciente con la disminución de la distancia entre y y o^L.

El algoritmo de aprendizaje por excelencia para redes multicapa es el de *gradiente descendente* que requiere el cálculo de los gradientes del error con respecto al peso y bias para la actualización de los mismos. Para este cálculo de gradientes se utiliza el **algoritmo de retropropagación** (o backpropagation, en inglés). Para formalizar este algoritmo se propone el diagrama de bloques generalizado de la Figura 4.5 para una red neuronal multicapa.

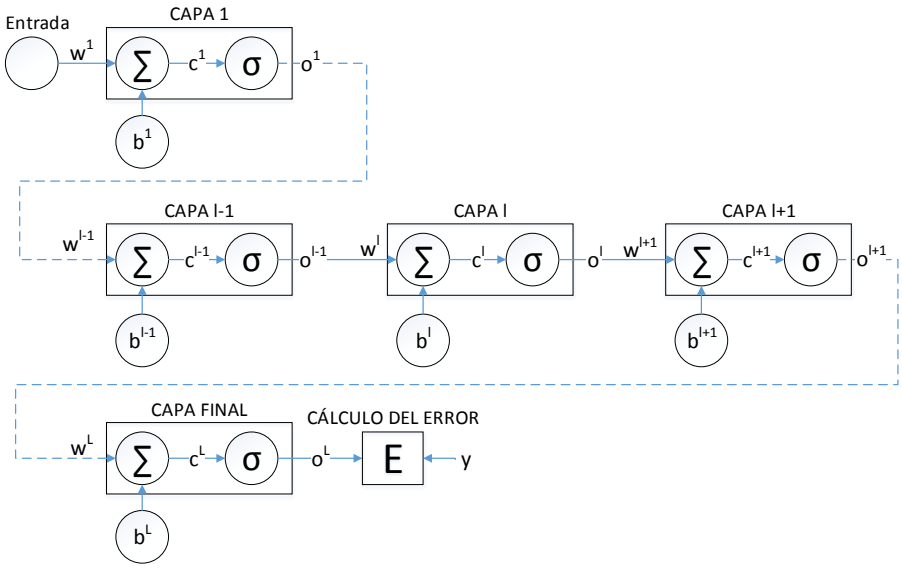


Figura 4.5: Diagrama de bloques generalizado de una red neuronal multicapa.

Para el diagrama presentado, la función de interacción c de las entradas con los pesos y el bias, y la salida o para una capa l viene dada por:

$$c^l = w^l o^{l-1} + b^l \tag{4.10}$$

$$o^l = \sigma(c^l) \tag{4.11}$$

Para optimizar los pesos y bias es necesario primero conocer la variación del error con una variación en el peso o bias. Esto se formaliza calculando las derivadas parciales $\frac{\partial E}{\partial w}$ y $\frac{\partial E}{\partial b}$. Para encontrar esta variación del error con respecto a los pesos de

una capa l, se plantea la siguiente resolución a partir de la regla de la cadena:

$$\frac{\partial E}{\partial w^l} = \frac{\partial E}{\partial c^l}\frac{\partial c^l}{\partial w^l} \tag{4.12}$$

$$\delta^l = \frac{\partial E}{\partial c^l} \tag{4.13}$$

$$\frac{\partial c^l}{\partial w^l} = o^{l-1} \tag{4.14}$$

$$\frac{\partial E}{\partial w^l} = \delta^l o^{l-1} \tag{4.15}$$

De igual manera, para encontrar la variación del error con respecto al bias se resuelve:

$$\frac{\partial E}{\partial b^l} = \frac{\partial E}{\partial c^l}\frac{\partial c^l}{\partial b^l} \tag{4.16}$$

$$\frac{\partial c^l}{\partial b^l} = 1 \tag{4.17}$$

$$\frac{\partial E}{\partial b^l} = \delta^l \tag{4.18}$$

El término δ^l que representa la variación del error con respecto a la función de las entradas de la capa actual, c^l, puede ser representado como una función de las entradas de la capa siguiente, permitiendo retro-propagar el error como se muestra a continuación:

$$\delta^l = \frac{\partial E}{\partial c^{l+1}}\frac{\partial c^{l+1}}{\partial c^l} \tag{4.19}$$

$$\frac{\partial E}{\partial c^{l+1}} = \delta^{l+1} \tag{4.20}$$

$$\frac{\partial c^{l+1}}{\partial c^l} = \frac{\partial}{\partial c^l}(w^{l+1}o^l + b^{l+1}) \tag{4.21}$$

$$= \frac{\partial}{\partial c^l}[w^{l+1}\sigma(c^l) + b^{l+1}] \tag{4.22}$$

$$= w^{l+1}\sigma'(c^l) \tag{4.23}$$

$$\delta^l = [w^{l+1}]^T\delta^{l+1} \odot \sigma'(c^l) \tag{4.24}$$

Finalmente, el cálculo del término δ para la capa L de la red (la capa final), que es donde empieza la retro-propagación del error viene dado por:

$$\delta^L = \frac{\partial E}{\partial o^L}\frac{\partial o^L}{\partial c^L} \tag{4.25}$$

$$= \frac{\partial E}{\partial o^L} \odot \sigma'(c^L) \tag{4.26}$$

De esta forma, el problema de aprendizaje de pesos (y bias) adecuados se ha traducido en un problema clásico de optimización de funciones derivables.

4.2.3. Red neuronal convolucional (CNN)

Una red convolucional es un modelo de aprendizaje bio-inspirado en el cortex visual de los cerebros animales. A partir de un conjunto de datos, este modelo busca aprender una colección óptima de kernels con respecto a una tarea específica.

Debido a las características que presentan, y que analizaremos a continuación, en su origen las CNNs fueron utilizadas principalmente en tareas de reconocimiento de imágenes. Por ejemplo, el modelo LeNet-5 propuesto en [47] representa uno de los primeros ejemplos de uso, y su objetivo era mejorar las capacidades de las redes neuronales en la tarea de reconocimiento de caracteres.

Las redes convolucionales están compuestas por un conjunto de capas apiladas de forma jerárquica que tienen la convolución como operación fundamental, donde la salida de una capa es la entrada de la siguiente. La convolución se aplica tomando como límites el tamaño $M \times N$ de la señal bidimensional de entrada y alineando con ellos al kernel de tamaño $m \times n$, y dando como resultado una nueva señal bidimensional de tamaño $M-m+1 \times N-n+1$, menor a la original. Sin embargo, se pueden aplicar otras técnicas como, por ejemplo, rellenar la señal de entrada con elementos adicionales o unir los extremos de la señal de entrada para formar un toroide, etc. Adicionalmente, al conjunto de capas de convolución se añade al final un MLP que conformará la etapa de clasificación de la red.

La arquitectura de una CNN con una capa convolucional se muestra en la Figura 4.6, en donde se puede apreciar el flujo de información desde que la señal bidimensional ingresa a la red hasta que se realiza una clasificación. Junto a la operación de convolución explicada, en este tipo de redes podemos encontrar dos operaciones adicionales, denominadas *sub-muestreo* (parte de la capa convolucional) y *aplanamiento de mapas de características* (previo a la etapa de clasificación) que explicaremos a continuación:

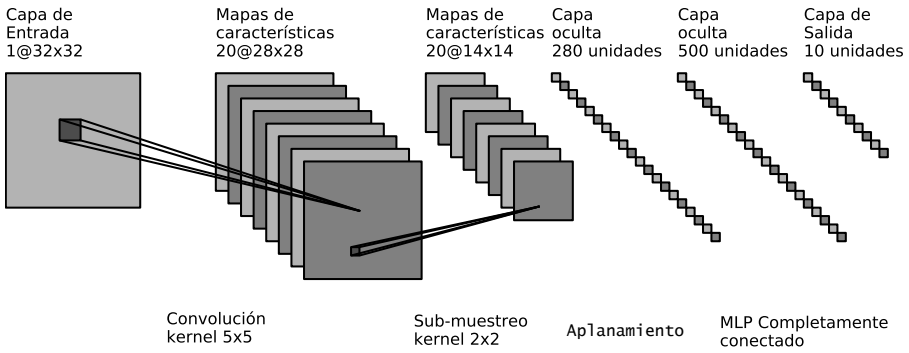


Figura 4.6: CNN con una capa convolucional.

Sub-muestreo El proceso de sub-muestreo es parte de la capa convolucional y tiene sus orígenes en el procesamiento de señales, en donde se utiliza para disminuir

la cantidad de información de una señal con el fin de adaptarla a una nueva tasa de muestreo menor a la original. En las CNN el sub-muestreo tiene un objetivo similar, es decir, busca disminuir de forma progresiva la masiva cantidad de datos presente en los mapas de características resultantes del proceso de convolución. Mediante este procedimiento se logra disminuir la complejidad computacional de la red y reducir el número de características extraídas.

El tipo de sub-muestreo más utilizado es el *max-pooling*, que desplaza una ventana de selección de elementos por el espacio de cada mapa de características sin solapamiento hasta completar su ancho y largo. Para cada posición de la ventana, se elige el valor mayor de los elementos seleccionados con los cuales se construye un nuevo mapa de características inversamente proporcional a la dimensión preestablecida de la ventana. Un ejemplo de max-pooling es presentado en la Figura 4.7 para un mapa de características de 4×4 y una ventana de 2×2. La capa de

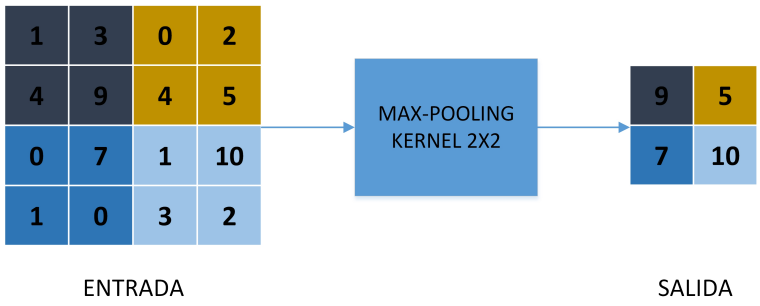


Figura 4.7: Capa de max-pooling aplicada a un mapa de características de 4×4.

sub-muestreo, además de disminuir la complejidad computacional de la red, proporciona un mecanismo de invarianza a pequeños desplazamientos (locales) de los patrones encontrados por la operación de convolución, lo cual brinda robustez adicional a la red. Sin embargo, el uso de la capa de sub-muestreo debe ser tratado con cautela porque se podrían perder características relevantes y afectar de este modo la sensibilidad de la red.

Aplanamiento de mapas de características Esta operación se aplica después del conjunto de capas convolucionales y sub-muestreo para dar inicio a la etapa de clasificación. Con la aplicación de cada capa convolucional (y opcionalmente sub-muestreo), se logran extraer patrones cada vez más independientes de la posición en la que aparecen en la señal de entrada. De esta forma, la capa convolucional más profunda devuelve un grupo de mapas de características con una activación robusta en sus elementos ante la variabilidad de la entrada a la red, y solamente cambiante con la presencia o ausencia de los diferentes patrones detectados por los filtros.

Este grupo de mapas de características es aplanado mediante conversiones sucesivas de matriz a vector para cada mapa y el posterior apilamiento de todos los vectores resultantes. Un ejemplo de esta operación se muestra en la Figura 4.8.

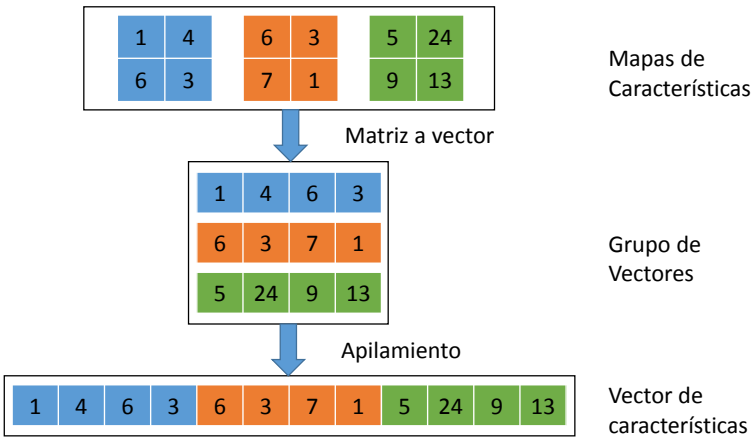


Figura 4.8: Ejemplo de aplanamiento de 3 mapas de características con tamaño 4×4.

Aprendizaje

De forma similar a como se entrenan las redes neuronales clásicas, el proceso de aprendizaje en una CNN consta de dos etapas:

- Propagación.
- Retro-propagación.

A continuación vamos a explicar brevemente cómo funciona cada una de estas etapas para este tipo de redes:

Propagación

Las entradas y salidas de una capa convolucional se pueden interpretar como un conjunto de funciones, en los casos que veremos, bidimensionales, que vamos a llamar *canales*. Concretamente, el m-ésimo elemento del conjunto de salida en la capa convolucional l, $o_m^l(x, y)$ viene determinado por:

$$o_m^l(x, y) = \sigma(c_m^l(x, y)), \text{ donde} \tag{4.27}$$

$$c_m^l(x, y) = \sum_n i_{n,m}^l(x, y) + b_m^l \tag{4.28}$$

$$i_{n,m}^l(x, y) = \sum_{x_1, y_1} o_n^{l-1}(x + N_w - 1 - x_1, y + N_w - 1 - y_1)\, w_{n,m}^l(x_1, y_1) \tag{4.29}$$

siendo $w_{n,m}^l(x_1, y_1)$ el m-ésimo kernel aplicado al n-ésimo canal de la entrada en la capa l, N_w es el tamaño del kernel, $b_m^l(x, y)$ es la m-ésima función de bias en la capa l, y σ alguna función no lineal (muy habitualmente $tanh$). La n-ésima función de entrada a la red está representada por $o_n^0(x, y)$ y, en general, cada función de entrada

a la capa convolucional l vendrá dada por la salida de la capa anterior, es decir, $o_n^{l-1}(x, y)$.

Retro-propagación

De acuerdo a lo visto en la propagación, en una capa convolucional la salida depende de los parámetros w y b, por lo que los esfuerzos del entrenamiento deben ir encaminados en optimizar la función de error modificando adecuadamente estos parámetros.

Siguiendo un razonamiento similar al que se sigue con el entrenamiento de redes neuronales clásicas, como el error de una red, está en función de la salida de la última capa, y ésta a su vez es una composición de funciones de las capas anteriores, la modificación de los parámetros asociados a una capa l, cualquiera, debe realizarse teniendo en cuenta su influencia en la capa $l+1$, que a su vez influirá en la capa $l+2$, y así sucesivamente hasta llegar a la capa última y al error medible. En concreto, este proceso iterativo de aproximación de los parámetros de la red para la minimización del error usando un procedimiento de gradiente descendente queda explicitado para el caso convolucional de la siguiente forma:

Derivada del error con respecto al kernel. Vamos a medir cómo influye cada parámetro $w_{n,m}^l(x, y)$ de una capa l (el kernel) en el error de la red completa, E. Esto es equivalente a la suma de la influencia de cada punto en c_m^l en E, ponderada por la influencia de $w_{n,m}(x, y)$ en cada punto concreto de c_m^l:

$$\frac{\partial E}{\partial w_{n,m}^l(x,y)} = \sum_{x_1,y_1} \frac{\partial E}{\partial c_m^l(x_1,y_1)} \frac{\partial c_m^l(x_1,y_1)}{\partial w_{n,m}^l(x,y)} \tag{4.30}$$

La influencia de cada punto en c_m^l en E, que de nuevo denotaremos por δ, es visto como el grado de cambio de E con respecto al cambio en la salida de una capa.

$$\delta_m^l(x_1,y_1) = \frac{\partial E}{\partial c_m^l(x_1,y_1)} \tag{4.31}$$

La influencia de $w_{n,m}(x, y)$ en cada punto concreto de c_m^l es obtenida desarrollando la derivada de la ecuación 4.28.

$$\frac{\partial c_m^l(x_1,y_1)}{\partial w_{n,m}^l(x,y)} = \frac{\partial}{\partial w_{n,m}^l(x,y)} \left(\sum_{n,x_2,y_2} o_n^{l-1}(\bar{x}(x_2), \bar{y}(y_2))\, w_{n,m}^l(x_2,y_2) + b_m^l \right) \tag{4.32}$$

$$\frac{\partial c_m^l(x_1,y_1)}{\partial w_{n,m}^l(x,y)} = o_n^{l-1}(\bar{x}(x), \bar{y}(y)) \tag{4.33}$$

donde $\bar{x}(x) = x_1 + N_w - 1 - x_2$, y $\bar{y}(y) = y_1 + N_w - 1 - y_2$.

Reemplazando la ecuación 4.31 y 4.33 en 4.30, obtenemos:

$$\frac{\partial E}{\partial w^l_{n,m}(x,y)} = \sum_{x_1,y_1} o^{l-1}_n(\bar{x}(x),\bar{y}(y))\ \delta^l_m(x_1,y_1) \tag{4.34}$$

Es necesario notar que la ecuación anterior puede ser reescrita como una operación de cross-correlación seguida por una rotación de $180°$ de la matriz resultante:

$$\frac{\partial E}{\partial w^l_{n,m}(x,y)} = rot_{180}(\sum_{x_1,y_1} o^{l-1}_n(x+x_1,y+y_1)\ \delta^l_m(x_1,y_1)) \tag{4.35}$$

Derivada del error con respecto al bias El mismo procedimiento se aplica para encontrar cómo afecta el parámetro b a E:

$$\frac{\partial E}{\partial b^l_m(x,y)} = \sum_{x_1,y_1} \delta^l_m(x_1,y_1)\ \frac{\partial c^l_m(x_1,y_1)}{\partial b^l_m(x,y)} \tag{4.36}$$

$$\frac{\partial c^l_m(x_1,y_1)}{\partial b^l_m(x,y)} = \frac{\partial(\sum_{n,x_2,y_2} o^{l-1}_n(\bar{x}(x_2),\bar{y}(y_2))\ w^l_{n,m}(x_2,y_2) + b^l_m)}{\partial b^l_m(x,y)} \tag{4.37}$$

En este caso todos los valores dentro del sumatorio son independientes de b, y por lo tanto su derivada será 0, entonces:

$$\frac{\partial c^l_m(x_1,y_1)}{\partial b^l_m(x,y)} = 1 \tag{4.38}$$

Reemplazando la ecuación 4.31 y 4.38 en 4.37, obtenemos:

$$\frac{\partial E}{\partial b^l_m(x,y)} = \sum_{x_1,y_1} \delta^l_m(x_1,y_1) \tag{4.39}$$

Cálculo de δ y retro-propagación del error Para el cálculo de δ en una capa hacemos uso nuevamente de la regla de la cadena. Entonces se expresa que la variación del m-ésimo canal resultante c^l_m afecta a la variación de E, como la suma de las afecciones del m-ésimo canal de la capa l en el o-ésimo canal de la capa $l+1$, ponderado por las afecciones del o-ésimo canal de $l+1$ en la variación de E.

$$\delta^l_m(x,y) = \sum_o \sum_{x_1,y_1}^{N_w-1,N_w-1} \frac{\partial E}{\partial c^{l+1}_o(x+x_1,y+y_1)}\ \frac{\partial c^{l+1}_o(x+x_1,y+y_1)}{\partial c^l_m(x,y)} \tag{4.40}$$

Notese aquí que una coordenada (x,y) de c^l_m afecta a N_w-1 coordenadas circundantes en su equivalente de la capa $l+1$, lo cual esta considerado en la ecuación dada.

El valor de δ es tomado de la capa siguiente y es el término que retro-propaga el error hacia las capas anteriores. De esta manera, el primer δ que se calcula es el de la capa final del red.

$$\delta_o^{l+1}(x + x_1, y + y_1) = \frac{\partial E}{\partial c_o^{l+1}(x + x_1, y + y_1)} \tag{4.41}$$

Las afecciones del m-ésimo canal de la capa l en el o-ésimo canal de la capa $l + 1$ vienen dadas por:

$$\frac{\partial c_o^{l+1}(x + x_1, y + y_1)}{\partial c_m^l(x, y)} = \frac{\partial}{\partial c_m^l(x, y)} \left[\sum_{m_1} \sum_{x_2, y_2}^{N_w - 1} o_{m_1}^l(\bar{x}, \bar{y})\, w_{m_1,o}^{l+1}(x_2, y_2) + b_o^{l+1} \right] \tag{4.42}$$

$$\frac{\partial c_o^{l+1}(x + x_1, y + y_1)}{\partial c_m^l(x, y)} = \frac{\partial}{\partial c_m^l(x, y)} \left[\sum_{m_1} \sum_{x_2, y_2}^{N_w - 1} \sigma(c_{m_1}^l(\bar{x}, \bar{y}))\, w_{m_1,o}^{l+1}(x_2, y_2) + b_o^{l+1} \right] \tag{4.43}$$

$$\frac{\partial c_o^{l+1}(x + x_1, y + y_1)}{\partial c_m^l(x, y)} = \frac{\partial}{\partial c_m^l(x, y)} \left[\sigma(c_m^l(x, y))\, w_{m,o}^{l+1}(x_1, y_1) \right] \tag{4.44}$$

$$\frac{\partial c_o^{l+1}(x + x_1, y + y_1)}{\partial c_m^l(x, y)} = w_{m,o}^{l+1}(x_1, y_1) \sigma'(c_m^l(x, y)) \tag{4.45}$$

donde $\bar{x} = x + x_1 - x_2$, y $\bar{y} = y + y_1 - y_2$.

Reemplazando la ecuación 4.41 y 4.45 en 4.40, obtenemos que:

$$\delta_m^l(x, y) = \left[\sum_o \sum_{x_1, y_1}^{N_w - 1} \delta_o^{l+1}(x + x_1, y + y_1)\, w_{m,o}^{l+1}(x_1, y_1) \right] \sigma'(c_m^l(x, y)) \tag{4.46}$$

Lo que, de nuevo, nos ha permitido expresar el procedimiento de aprendizaje de parámetros de la red como un proceso de optimización de una función derivable haciendo uso de mecanismos analíticos.

4.2.4. Autoencoder

Un Autoencoder (AE), presentado por primera vez en [86] y utilizado por Geoffrey Hinton en [36], es una red que replica en la capa de salida los datos que ingresan en la capa de entrada. Su principal utilidad radica en su capacidad de aprender una mejor representación del espacio de entrada de una forma no-supervisada, esto es, sin el uso de etiquetas conocidas a priori. Esta tarea se realiza buscando una codificación desde la cual sea posible reconstruir los datos de entrada como se muestra en la Figura 4.9. Es decir, en el autoencoder se intenta que los datos de entrada y salida sean lo más similares posible.

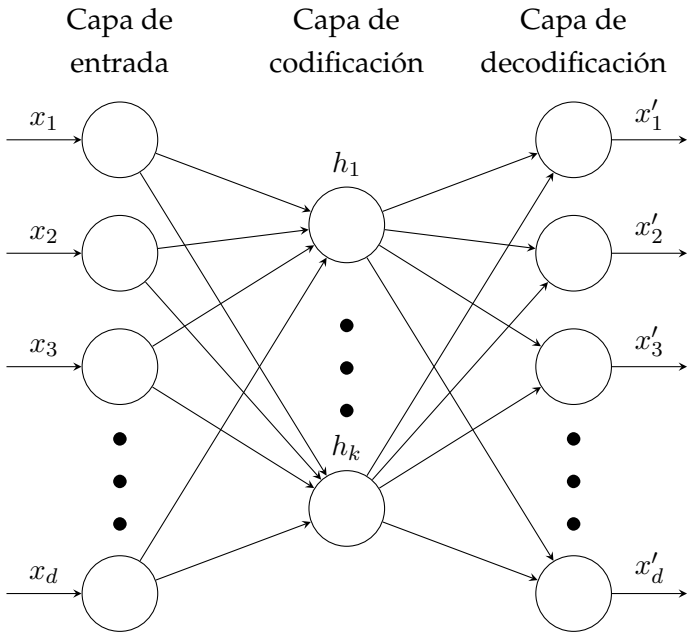


Figura 4.9: Un Autoencoder de tres capas.

Autoencoder clásico

En la fase de codificación, un vector de entrada $\mathbf{x} \in \mathbb{R}^d$ es normalmente codificado a un vector $\mathbf{h} \in [-1, 1]^k$ mediante el mapeo:

$$\mathbf{h} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{4.47}$$

donde $\mathbf{W} \in \mathbb{R}^{k \times d}$ y $\mathbf{b} \in \mathbb{R}^k$ son los parámetros que se pueden entrenar del codificador. Si $f(\cdot)$ es una función lineal, entonces el proceso de codificación es equivalente al Análisis de Componentes Principales[2] con una descomposición en k componentes principales, donde cada componente es la proyección del vector de entrada $\mathbf{x}$ en el espacio $\mathbb{R}^k$ [43]. Por otro lado, si $f(\cdot)$ es una función no-lineal entonces $\mathbf{x}$ no puede ser representado como una combinación lineal y el proceso se aleja de una descomposición por PCA. En muchas ocasiones, una función no-lineal permite capturar patrones que están ocultos para funciones lineales.

Tras el proceso de codificación, el vector de representación h es decodificado con un mapeo similar:

$$\mathbf{x}' = f(\mathbf{W}'\mathbf{h} + \mathbf{b}') \tag{4.48}$$

donde $x' \in \mathbb{R}^d$ es la reconstrucción del vector de entrada, $\mathbf{W}' \in \mathbb{R}^{d \times k}$ y $\mathbf{b}' \in \mathbb{R}^d$ son los parámetros del decodificador.

[2]Mejor conocido como PCA, por sus siglas en inglés.

El proceso de entrenamiento de un autoencoder puede resumirse como el proceso de minimización del error entre $\mathbf{x}$ y $\mathbf{x}'$, por ejemplo:

$$E = ||\mathbf{x} - \mathbf{x}'||_2^2 \tag{4.49}$$

Si se restringe $\mathbf{W}'$ a $\mathbf{W}' = \mathbf{W}^\top$, el problema de optimización se reduce a encontrar $\mathbf{W}$, $\mathbf{b}$ y $\mathbf{b}'$ que minimicen la función de error:

$$\mathbf{W}, \mathbf{b}, \mathbf{b}' = \underset{\mathbf{W},\mathbf{b},\mathbf{b}'}{\arg\min}\, E \tag{4.50}$$

y que, de nuevo, puede ser resuelto mediante el algoritmo de retro-propagación con gradiente descendente estocástico[3] [5].

Denoising autoencoder

La concepción original del AE establece que la dimensión del espacio de codificación (k) debe ser menor que la del espacio de entrada (d), en caso contrario se corre el peligro de aprender la función identidad, lo cual eliminaría cualquier capacidad de generalización del modelo.

Una técnica para evitar el aprendizaje de la función trivial es el uso de una versión estocástica de autoencoder llamada *Denoising Autoencoder* (dAE) [98], que intenta reconstruir el vector de entrada $\mathbf{x}$ desde una versión corrupta $\tilde{\mathbf{x}}$. La hipótesis que inspira esta idea expresa que una buena representación, $\mathbf{h}$, debe captura relaciones y dependencias entre los elementos de $\mathbf{x}$, y estos patrones son robustos y pueden ser obtenidos desde una versión parcialmente destruida del mismo, $\tilde{\mathbf{x}}$, por lo que sería posible reconstruir $\mathbf{x}$ desde una representación $\mathbf{h}$ que fue generada desde $\tilde{\mathbf{x}}$.

El proceso para entrenar un dAE empieza con una degeneración estocástica de la entrada $\mathbf{x}$, algo que podría ser realizado mediante cualquier proceso de destrucción aleatoria. El enfoque más común es la generación de un vector $\mathbf{r} \in \{0,1\}^d$, sujeto a una distribución de Bernoulli. Posteriormente, $\tilde{\mathbf{x}}$ es construido mediante $\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{r}$.

La entrada degradada $\tilde{\mathbf{x}}$ es codificada a $\mathbf{h}$ usando la ecuación (4.47). Luego $\mathbf{h}$ es decodificado a $\mathbf{x}'$ mediante la ecuación (4.48). El proceso de optimización de los parámetros de la red se realiza como en el autoencoder clásico.

4.2.5. Autoencoder convolucional

El autoencoder, en cualquiera de sus formas (clásica, denoising, etc), es un modelo que permite el aprendizaje no-supervisado de relaciones y dependencias entre los datos de entrada. Esto quiere decir que la codificación $\mathbf{h}$ podría ser una representación más compacta y clara de los datos de entrada, logrando obtener una nivel de abstracción superior de los datos de forma independiente de la tarea que se desea

[3] SGD por sus siglas en inglés.

realizar (clasificación, regresión, clustering, etc). Este resultado puede ser utilizado para crear redes de múltiples capas apiladas una después de otra. Cada capa será pre-inicializada de forma no-supervisada mediante un autoencoder. El objetivo es conseguir niveles de abstracción superiores como lo ha demostrado Hinton en [35].

La idea del autoencoder puede ser extrapolada para la inicialización de otros tipos de capas en una red neuronal, como las convolucionales en una CNN. Sin embargo, el autoencoder en su forma original no puede ser utilizado para pre-inicializar capas de una red convolucional, ya que ambos modelos poseen una arquitectura distinta[4].

Una modificación de la idea original de la red autoencoder para que pueda ser aplicada a capas convolucionales es presentada por Masci en [68] con el nombre de Autoencoder Convolucional (CAE por sus siglas en inglés), donde el modelo es usado concretamente para la extracción no supervisada de características para la base de datos MNIST de dígitos escritos a mano (una tarea de reconocimiento desde un gran conjunto de datos de imágenes)[48].

El objetivo del CAE es inicializar una capa convolucional con una mejor estimación de los parámetros $w^l_{n,m}$ y b^l_m. La arquitectura de este modelo para una capa convolucional l se muestra en la Figura 4.10.

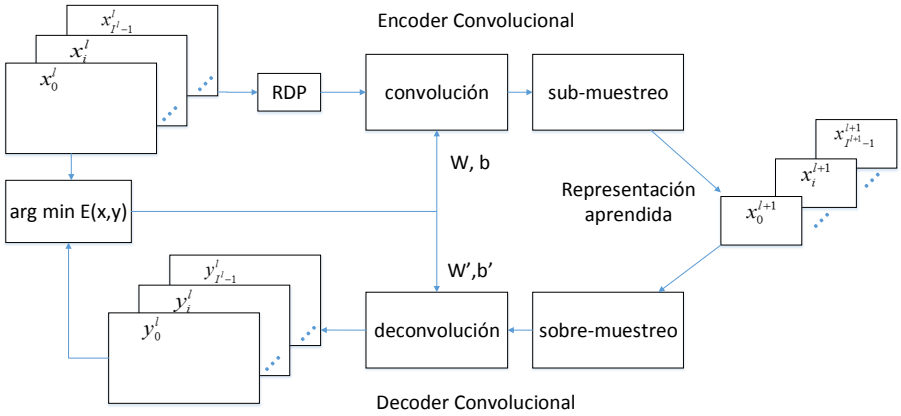


Figura 4.10: Arquitectura de CAE aplicado a una capa convolucional l.

De forma general, CAE imita el comportamiento de autocodificación de AE para datos bidimensionales. Las etapas de este modelo son descritas a continuación:

1. **RDP (Random Degeneration Process):** hace referencia al proceso de corrupción de los datos de entrada que evita el aprendizaje de la función trivial. Esta operación es similar a la usada en dAE pero aplicada en este caso a los mapas de características de entrada. En este sentido, el tipo de degradación elegida dependerá del tipo de dato de cada pixel, p, en los mapas de características.

[4]Nótese la diferencia entre la arquitectura clásica de red neuronal mostrada para el autoencoder en la Figura 4.9 con la estructura bidimensional requerida para la entrada a una CNN en la Figura 4.6.

Así, si $p \in \{0, 1\}$ (imagen a blanco y negro) se elige una degradación r por ruido impulsivo, o más conocido como sal y pimienta. Por otro lado, si $p \in \mathbb{R}$ (imagen a escala de grises) se elige una degradación r por ruido gaussiano. En cualquiera de los dos casos los mapas de características resultantes de este proceso se obtienen con:

$$\hat{x}_i^l = x_i^l \odot r \quad \forall i \in [0, I^l - 1] \tag{4.51}$$

donde I^l es el número de mapas de características para la capa l.

2. **Convolución:** La fase de convolución viene dada siguiendo la ecuación (4.29), la ecuación (4.28) y la ecuación (4.27), típicas de una capa convolucional de una CNN.

3. **Sub-muestreo:** El sub-muestreo, como se vió en las CNN, hace referencia a cualquier estrategia que permita reducir la cantidad de datos manteniendo la información significativa. Hasta esta etapa, el procedimiento realizado ha sido similar al efectuado en una capa convolucional clásica con la adición de RDP. Como resultado se obtiene el conjunto de mapas de características de salida, x_i^{l+1} $\forall i \in [0, I^{l+1} - 1]$, con información acorde a W y b que pueden ser vistos como componentes base de los mapas de entrada, los cuales además tienen un tamaño disminuido en un factor determinado por el max-pooling (reducción de dimensión). Este resultado puede interpretarse desde la teoría de AE como una codificación de la entrada. Posteriormente habrá de realizarse el proceso con el cual se recuperará la entrada a partir de esta codificación.

4. **Sobre-muestreo:** El propósito del sobre-muestreo es incrementar el tamaño de los datos. Habitualmente, para un conjunto de mapas de características esto se consigue replicando cada pixel en su vecindario un número determinado de veces igual al mismo factor del max-pooling (lo que se conoce como unpooling). Es necesario mencionar que max-pooling, y en general cualquier submuestreo, son operaciones destructivas, es decir, no se pueden recuperar los datos originales tras aplicar la operación. Esto indica que al aplicar el sobremuestreo no se garantiza de ninguna manera obtener los datos previos al submuestreo, solamente es una aproximación con igual dimensión.

5. **Deconvolución:** Esta operación intenta anular el efecto de la convolución aplicada anteriormente. Como es bien conocido en la teoría del procesamiento de señales, una forma de lograrlo es mediante una nueva operación de convolución que hace uso de las transpuestas, W' y b', de los kernels y bias originales W y b, obteniendo de esta forma el conjunto de características y_i^l, $\forall i \in [0, I^l - 1]$.

6. **Optimización:** Todo el proceso descrito anteriormente tiene como elementos clave (y los únicos que pueden ser aprendidos) W y b. Estos dos conjuntos de parámetros, que describen a la capa l, son optimizados de tal forma que disminuyan el error de reconstrucción entre los mapas x^l y y^l con la siguiente

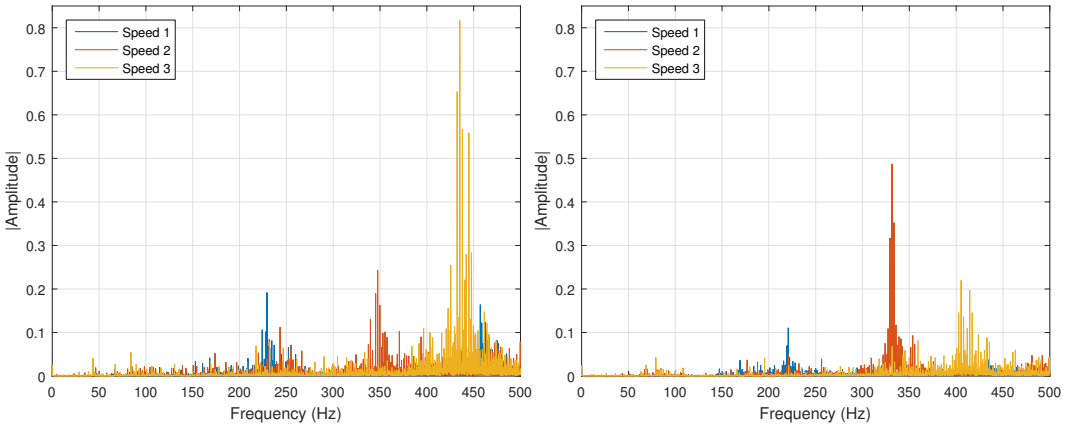
métrica (MSE para 2D):

$$E(W,b) = \frac{1}{2I^l} \sum_{i=0}^{I^l-1} (x_i^l - y_i^l)^2 \tag{4.52}$$

El procedimiento anterior hace siempre referencia a los elementos de una capa convolucional l. Cuando los parámetros se han optimizado para esta capa, puede continuarse a la capa siguiente, $l+1$, con mapas de entrada y obteniendo una estructura de CAEs apilados denominada SCAE. Esta arquitectura permite la optimización capa a capa de una red convolucional profunda.

4.3. Aprendizaje de características

El problema del descubrimiento de patrones que caractericen los estados en un sistema dinámico puede ser tan complejo como el sistema en sí mismo. Por ejemplo, en una maquinaria rotativa existen dificultades inherentes a la dependencia de las señales adquiridas con los parámetros de operación de la maquinaria, tal y como se ilustra en la Figura 4.11 con diferentes parámetros de operación estacionarios, donde el espectro de frecuencia de una señal de vibración muestra cambios en posición y forma a velocidades diferentes. Según muestran las mediciones reflejadas en esa Figura, los armónicos a Velocidad 1 se pueden confundir fácilmente con la frecuencia fundamental a Velocidad 3, algo común que sucede cuando las velocidades de operación son múltiplos entre sí. Aunque es cierto que las posiciones del espectro a diferentes cargas son muy similares, se aprecia que su forma cambia, aunque las velocidades sean las mismas en todos los casos (véase Figura 4.11a vs Figura 4.11b).



(a) Prueba para carga 1

(b) Prueba para carga 2

Figura 4.11: Espectro de frecuencia de una señal de vibración adquirida desde una caja de engranes helicoidales a tres diferentes velocidades constantes.

Bajo parámetros de operación no estacionarios, la identificación de patrones es aun más complicada. La Figura 4.12 ilustra el espectro a velocidad variable y diferentes cargas constantes. Para este caso, dependiendo de la carga, los patrones en el dominio de frecuencia no son identificables.

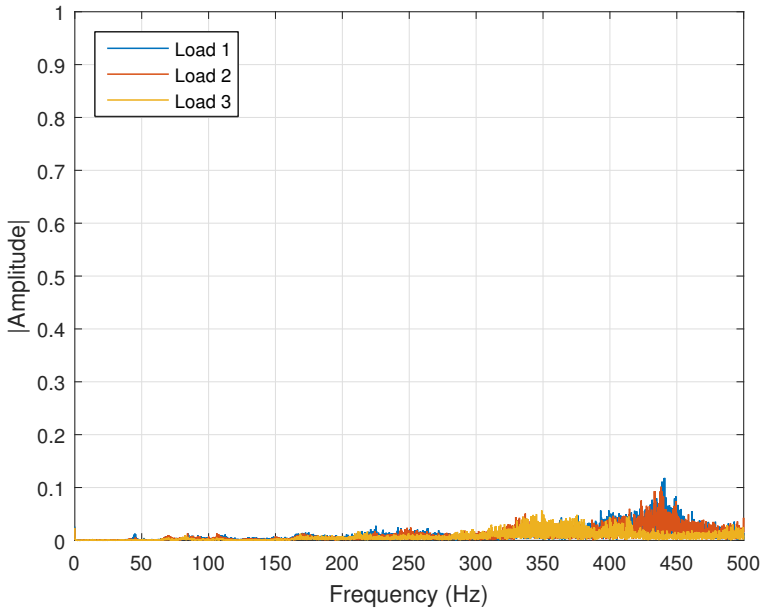


Figura 4.12: Espectro de frecuencia de la señal de vibración adquirida desde una caja de engranes helicoidales con velocidad variable y tres diferentes cargas constantes.

Para orientar el problema de identificación de patrones en el diagnóstico de maquinaria compuesta por engranes se han usado dos enfoques principales. El primero se basa en el análisis y metodologías para el procesamiento de señales, y tiene como propósito identificar bandas de frecuencias características en el espectro de las señales de vibración [27]. Es el enfoque más utilizado para el caso estacionario donde la localización de la banda suele indicar el tipo de fallo y su amplitud muestra la severidad. En el caso no estacionario, se han aplicado técnicas ad-hoc basadas en filtros para extraer una señal más informativa y así poder luego continuar con su análisis [53].

El segundo enfoque está basado en la clasificación de patrones de características extraídas de forma clásica desde las señales de vibración en el dominio del tiempo, frecuencia, y/o tiempo-frecuencia, como se muestra en la Figura 4.13. Cada característica es cuidadosamente diseñada dependiendo del caso de estudio para así lograr la extracción de los indicadores de condición más robustos, invariantes a cambios del proceso, e informativos. Tras el proceso de extracción de características se utilizan modelos clásicos de aprendizaje para la clasificación de patrones, como se muestra en [14, 77]. También se han aplicado algunos modelos de Deep Learning (por ejemplo, [55]) pero, al igual que en el caso anterior, estos modelos se suelen centrar en la clasificación clásica de patrones de características extraídas previamente.

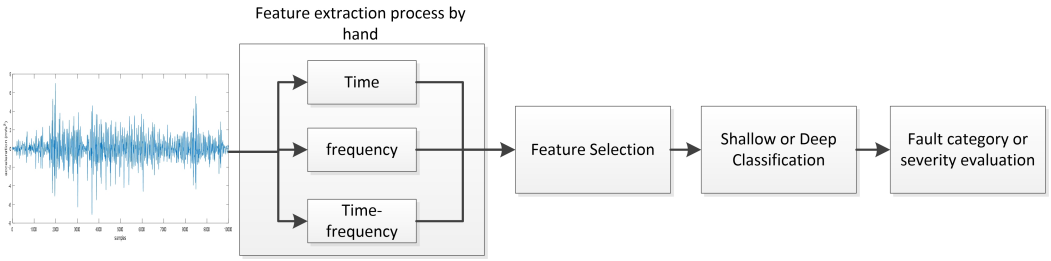


Figura 4.13: Metodología clásica para el diagnóstico de fallos mediante extracción de características convencional.

Como se ha notado anteriormente, la extracción y selección de características no son tareas triviales y son altamente dependientes de los parámetros operacionales. Debido a su complejidad, en este trabajo se propone un método para la evaluación de la severidad de fallos mediante la extracción automática de características desde series de tiempo medidas en una maquinaria bajo condiciones de operación estacionarias y no estacionarias. El método propuesto, representado en la Figura 4.14, está compuesto por los siguientes pasos:

1. **Adquisición de la Señal:** Adquirir las señales de vibración bajo múltiples valores de severidad de fallo con condiciones de operación estacionarias y no estacionarias.

2. **Representación en tiempo-frecuencia:** Transformar las señales de vibración del dominio de tiempo a su representación equivalente en tiempo-frecuencia.

3. **Extracción automática de características:** Construir el modelo DCNN inicializado por SCAE para la extracción de características, usando la representación en tiempo-frecuencia de las señales.

4. **Clasificación y regresión:** Con las características extraídas, construir el modelo de clasificación para estimar el nivel discreto de severidad de fallo. De forma opcional, agregar un capa de regresión de suma ponderada para obtener el valor continuo de severidad de fallo.

5. **Evaluación en línea de la severidad de fallo:** Evaluar la severidad del daño en un elemento mecánico a partir de una nueva medición de vibración usando los modelos obtenidos. La salida puede ser un nivel de severidad categórico o un valor de severidad en porcentaje.

Estas fases son detalladas en las siguientes subsecciones.

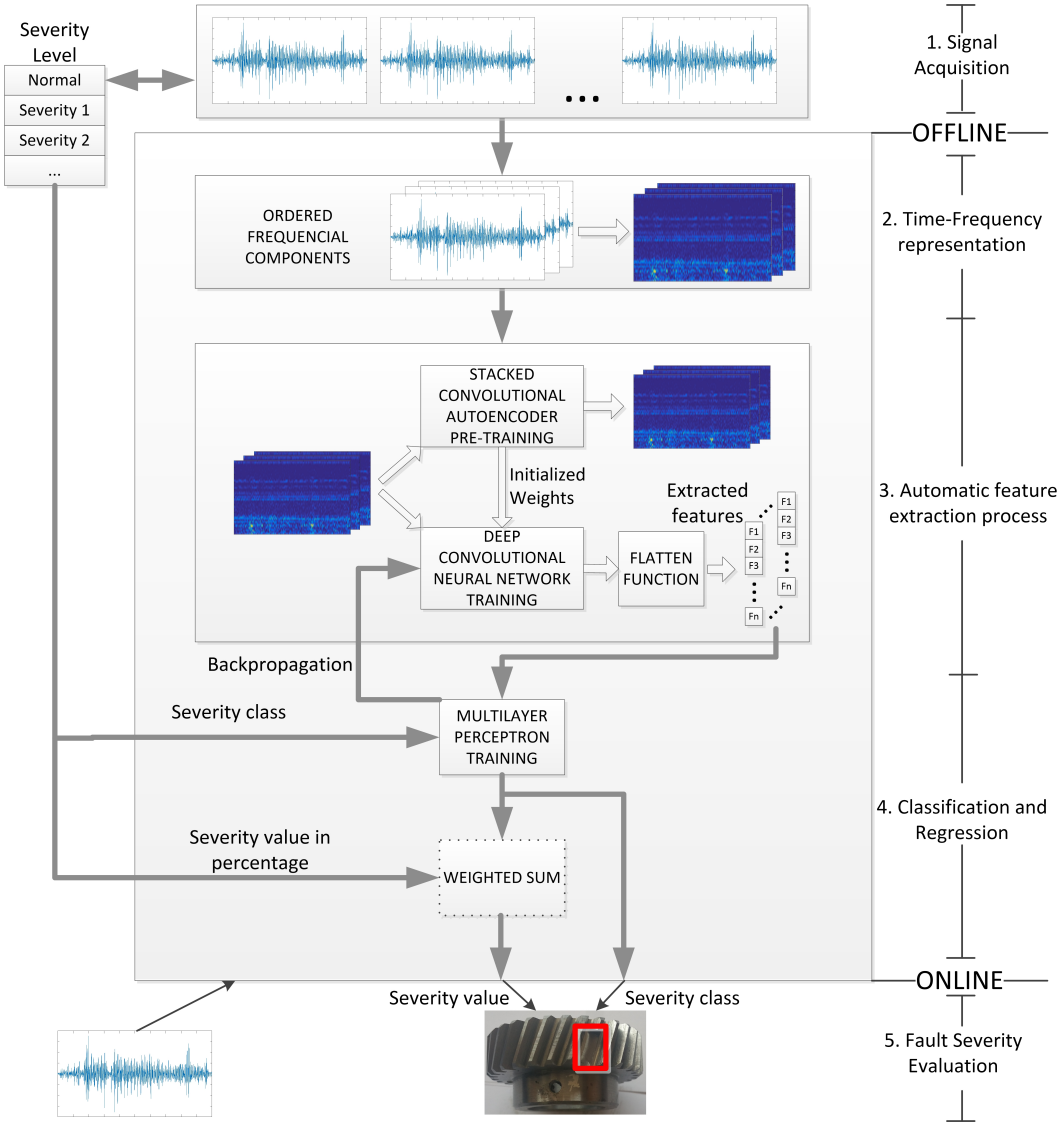


Figura 4.14: Metodología clásica para el diagnóstico de fallos mediante extracción de características convencional.

4.3.1. Adquisición de señales

Las señales de vibración se obtienen por medición directa de esta variable desde la maquinaria y su discretización se realiza por medio de un proceso de conversión analógico a digital. Cada señal de vibración es normalizada en el intervalo $[0, 1]$ y agrupada con otras señales de vibración bajo parámetros de operación de velocidad y carga diferentes o, en su defecto, no estacionarios.

4.3.2. Representación en tiempo-frecuencia

La representación de una señal en el dominio del tiempo puede ser transformada al dominio de tiempo-frecuencia de múltiples formas, entre las que destacan:

- Transformada corta de Fourier.
- Transformada fraccional de Fourier.
- Descomposición por Wavelet Packet (WPD).

Para el presente trabajo se usará WPD[31] debido que es una técnica que ofrece un ajuste de resolución adaptativa (multi-resolución) en el dominio del tiempo y frecuencia (véase Sección 2.4), algo necesario al trabajar bajo condiciones de operación de velocidad variable.

WPD se obtiene por la descomposición recursiva de la señal de ingreso mediante dos procesos paralelos:

$$A = s * g \tag{4.53}$$

$$D = s * h \tag{4.54}$$

donde s es la señal de entrada, g la respuesta al impulso de la función de wavelet padre ϕ (normalmente representada por un filtro pasa-bajo), h la respuesta al impulso de la función de wavelet madre, ψ (normalmente representada por un filtro pasa-alto), y A y D son las señales resultantes del proceso de descomposición (llamadas coeficientes de aproximaciones y detalles, respectivamente). Los dos filtros g y h se restringen a ser mutuamente complementarios. Las funciones de Wavelet pueden ser cualquier función con energía finita.

Las señales A y D se descomponen de forma recursiva con el fin de alcanzar un nivel deseable de descomposición, obteniendo de esta manera un árbol binario de descomposición de señales como se muestra en la Figura 4.15. El último nivel de los nodos tiene toda la información en los dominios de tiempo y frecuencia de la señal de entrada, y cada nodo en este nivel representa un rango específico en el espectro. Las señales dentro de estos nodos pueden ser normalizadas en la escala de tiempo y luego ser apiladas desde la frecuencia menor a la mayor. Mediante este procedimiento se obtiene la representación de tiempo-frecuencia mostrada en la Figura 4.16a y

su equivalente 2D en la Figura 4.16b. Como ejemplo de posibles patrones para estimación de la severidad del daño se han marcado los coeficientes con los valores más elevados. En un caso real los patrones serán más complejos y más difíciles de identificar.

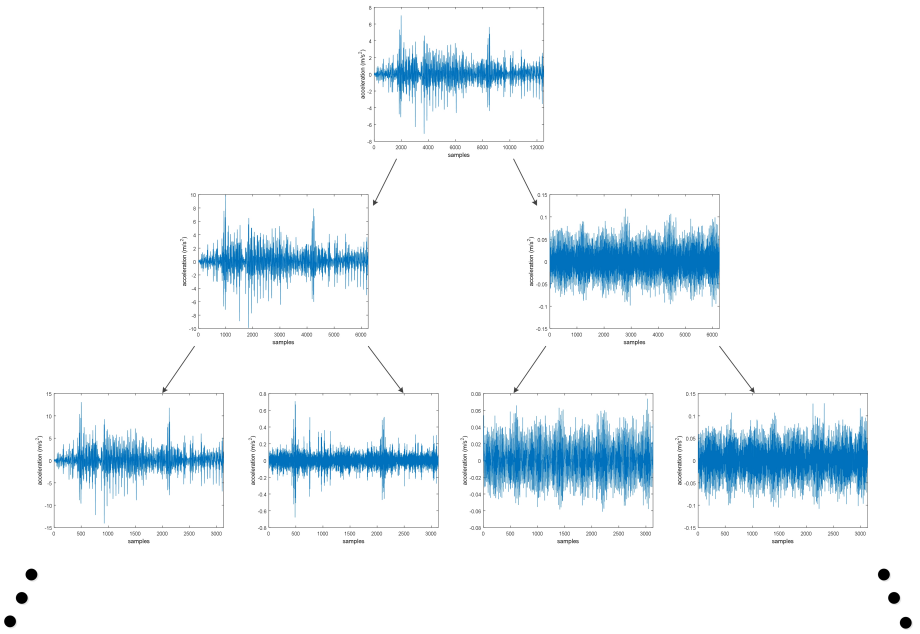


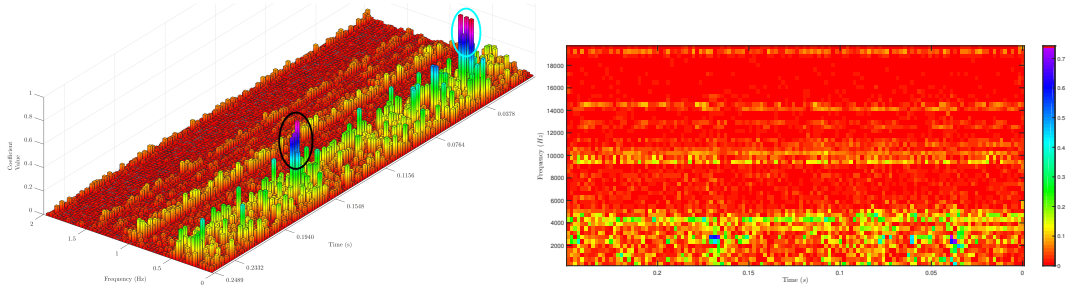
Figura 4.15: Árbol de descomposición por Wavelet Packet de una señal de vibración en el dominio del tiempo.

4.3.3. Proceso de extracción automático de características

Las características en la representación tiempo-frecuencia de una señal de vibración exhiben patrones altamente móviles debido a la carencia de una señal de sincronización externa por vuelta del motor. Esta característica requiere que el modelo reconozca características en cualquier localización de la representación tiempo-frecuencia.

El modelo de red convolucional profunda es capaz de detectar patrones relacionados, de forma local, con otros en una entrada bidimensional, independiente de la zona donde ocurren estos eventos. Esto significa que DCNN es robusta al desplazamiento y podría ser un buen candidato como modelo de extracción de características desde patrones con ocurrencia local en la representación tiempo-frecuencia. Pero, como éste es entrenado por el algoritmo de retropropagación desde parámetros inicializados de forma aleatoria, podría sufrir de convergencia prematura en los parámetros de las primeras capas por el problema de pérdida del gradiente.

La debilidad del modelo DCNN puede ser direccionada con un mejor método



(a) Representación tiempo-frecuencia en 3D (b) Representación tiempo-frecuencia en 2D

Figura 4.16: Representación en tiempo-frecuencia de la señal de vibración usando wavelet daubechies 5. La Figura 4.16a ilustra el apilamiento de las señales de los coeficientes de descomposición, donde la amplitud de cada barra representa el valor de los coeficientes en un tiempo y frecuencia específicos. La Figura 4.16b presenta la versión en 2D. Los óvalos marcan patrones equivalentes en la visualización en 3D y 2D con los colores acorde a su posición.

para inicializar sus parámetros. En este trabajo se propone el uso de la arquitectura Stacked Convolutional Autoencoders, que como hemos indicado está compuesta por CAE apilados, de forma similar a SdAE, con el fin de extraer un conocimiento a priori usando como métrica la capacidad del modelo para la reconstrucción de la representación tiempo-frecuencia de la señal de ingreso. Posteriormente, los parámetros obtenidos con SCAE son usados como punto de partida para un proceso de ajuste fino de la DCNN.

Cada elemento en el conjunto de mapas de características de la salida de la DCNN es pasado por una función de aplanamiento y apilado para construir un vector de características extraídas. El tamaño de este vector depende de la medida de los datos de entrada a la DCNN, donde la salida de cada capa es una colección de k mapas de características de tamaño $(\frac{M-m+1}{2} \times \frac{N-n+1}{2})$ cuando recibe como entrada mapas de características con tamaño $(M \times N)$ y se usan k kernels de tamaño $(m \times n)$.

4.3.4. Clasificación y Regresión

El vector de características obtenido mediante el proceso de extracción automático de características desde la representación en tiempo-frecuencia es usado como entrada para un perceptrón multi-capa (MLP), construido para evaluar la severidad del fallo. Tomando en cuenta que normalmente el espacio de decisión creado por el vector de características es de dimensión alta, el número de neuronas en la capa oculta del MLP debería ser suficientemente grande. El número de salidas para esta arquitectura es igual al número de valores de severidad disponibles establecido en la fase de entrenamiento.

El error de clasificación es calculado usando la clase correspondiente al nivel real

de severidad y la salida de la DCNN. Después, el error es retro-propagado desde la salida hacia las capas ocultas de la DCNN para completar el ciclo de ajuste fino en el proceso de extracción automática de características, con lo cual se logra ajustar las características para obtener una mejor precisión acorde a la estimación de la clase de severidad concreta.

Adicionalmente, de puede agregar una capa de regresión compuesta por una operación de suma ponderada (neurona lineal) de las entradas para obtener una salida de tipo continua. Para este propósito se puede mantener la capa de discretización tras el regresor logístico o eliminarla totalmente. En el primer caso, la clase estimada es directamente ponderada por su equivalente en porcentaje de daño. En el segundo caso, cada probabilidad de pertenencia a una clase es ponderada por su valor correspondiente de severidad en porcentaje y sumada con el resto de probabilidades ponderadas.

4.3.5. Evaluación en línea de la severidad del fallo

Tras el ajuste de los modelos siguiendo los pasos descritos anteriormente, se pueden realizar pruebas en línea. Para ello, se ingresa una nueva señal de vibración adquirida desde la maquinaria en la arquitectura de caja negra sin cambios en los parámetros internos. El proceso puede generar dos salidas: una primera con una estimación de tipo categórica de la severidad de daño, y una segunda con una estimación del porcentaje de daño presente en el elemento de la maquinaria evaluado.

4.4. Experimentación

Se han realizado tres grupos de pruebas, cada uno con un propósito específico. El primer grupo permite comparar entre la DCNN con kernels inicializados de forma aleatoria y la DCNN propuesta con kernels inicializados usando SCAE, evaluando la ventaja de iniciar el proceso de entrenamiento desde una solución pre-optimizada. Posteriormente, el mejor DCNN con SCAE fue evaluado en cada nivel de severidad para analizar si la mayoría de los errores se producen entre clases adyacentes.

El segundo grupo fue realizado con el propósito de evaluar la incidencia de usar el regresor lineal con la capa de discretización en el modelo del MLP. El objetivo fue medir si el conjunto de probabilidades dan información en concordancia con el nivel de severidad continuo y si cada probabilidad es cercana al resto.

El tercer grupo proporciona una comparación con otros métodos, supervisados y no-supervisados, que han sido aplicados previamente a casos de diagnóstico de fallos similares. Los métodos seleccionados usan técnicas clásicas de extracción de características sin el diseño convencional de características para, de esta manera, poder obtener una comparación entre métodos que buscan el mismo objetivo: la automatización en el proceso de extracción de características desde series de tiempo.

Código	kernels en capa 1	kernels en capa 2	kernels en capa 3
CNN1	100 × (5,5)	200 × (3,3)	-
CNN2	100 × (5,5)	40 × (5,5)	-
CNN3	100 × (5,5)	200 × (3,3)	400 × (3,3)

Cuadro 4.1: Configuración de cada CNN con el número de kernels y el tamaño para cada kernel.

Los detalles de estas pruebas son presentados en las subsecciones siguientes.

4.4.1. Comparación DCNN vs SCAE-DCNN

La primera prueba fue realizada usando DCNN con inicialización aleatoria de parámetros, uniformemente muestreada desde un rango acotado. Se probaron arquitecturas con 2 y 3 capas convolucionales. Los factores comunes tanto en las arquitecturas utilizadas como en los parámetros de entrenamiento son:

- Presencia de una capa de max-pooling en cada capa convolucional con un factor de 2.
- Una única capa oculta de 1000 neuronas completamente conectadas a la última capa convolucional.
- Capa final formada por un regresor logístico.
- 200 elementos en cada minibatch.
- Tasa de aprendizaje para entrenamiento supervisado de 0.1.
- 400 épocas de entrenamiento.

Las configuraciones evaluadas con diferentes número de kernels y su tamaño se resumen en la Tabla 4.1.

Usando esta configuración de arquitectura y parámetros de entrenamiento se realizaron otras pruebas con SCAE. En estas pruebas, la fase de pre-entrenamiento fue agregada con un nivel de corrupción de 0.1, 0.2 y 0.3 en la capa 1,2 y 3 respectivamente; y se usó un nivel de corrupción de 0.3 en la capa completamente conectada en todos los casos. La tasa de aprendizaje para el pre-entrenamiento fue de 0.01 y el número de épocas ajustado a 100. La Tabla 4.2 resume la arquitectura para cada SCAE.

La exactitud fue usada como métrica para cuantificar el desempeño global con DCNN y SCAE. En la Tabla 4.3 se muestra pequeñas diferencias entre CNN1 y CNN2, comparadas con SCAE1 y SCAE2, de 0.0015 entre CNN1 y SCAE1, y 0.0012

Código	kernels en capa 1	kernels en capa 2	kernels en capa 3
SCAE1	100 × (5,5)	200 × (3,3)	-
SCAE2	100 × (5,5)	40 × (5,5)	-
SCAE3	100 × (5,5)	200 × (3,3)	400 × (3,3)

Cuadro 4.2: Configuración de cada SCAEs con el número de kernels y el tamaño para cada kernel.

Code	CNN	SCAE
1	0.9208	0.919
2	0.9382	0.937
3	0.9242	0.9465

Cuadro 4.3: Exactitud de DCNN y SCAE sobre el conjunto de prueba.

entre CNN2 y SCAE2, evidenciando un ligero mejor desempeño de CNNs sobre SCAEs que casi podría ser considerado despreciable. Por otro lado, SCAE3 muestra un incremento significativo de desempeño de 0.0223 comparado a CNN3 (que tienen la misma arquitectura), y un incremento de 0.0083 comparado a CNN2, siendo este último el mejor de todos los CNNs.

SCAE3, que presenta la mejor exactitud, es evaluada en su desempeño para cada nivel de severidad usando las métricas clásicas: Precisión, Recall y valor-F, como se muestra en la Tabla 4.4. Todas estas métricas fueron calculadas a partir de la matriz de confusión mostrada en la Tabla 4.5, que fue obtenida al evaluar SCAE3 con el conjunto de prueba de 600 elementos por cada nivel de severidad.

La mejor precisión, que indica la capacidad del modelo para evitar la inclusión de elementos desde cualquier otra clase en la clase analizada, se obtiene de P1 (condición normal). Esto muestra que las firmas P2 a P10 tienen las mejores características de separabilidad con respecto a P1. El peor caso se obtiene en la confusión de P6 con las firmas P3 (10 elementos), P5 (11 elementos), y P7 (14 elementos).

Por otro lado, el mejor recall, que muestra la capacidad del modelo para incluir todos los elementos que, de hecho, están dentro de una clase, se obtiene en P8. Esto muestra que P8 es la firma más distinguible del resto de firmas. Nuevamente, el peor caso se obtiene para P6, que muestra su mayor error con P5 (7 elementos) y P7 (11 elementos), lo que puede estar justificado por la cercanía en el nivel de severidad de P6 con P5 y P7.

Adicionalmente, el valor-F muestra el desempeño general del modelo, considerando ambos, precisión y recall. En este caso, el mejor valor-F se obtiene para P8, y el peor para P6.

Código de Severidad	Precisión	Recall	Valor-F
P1	0.9680	0.9583	0.9631
P2	0.9533	0.9533	0.9533
P3	0.9538	0.9283	0.9409
P4	0.9400	0.9400	0.9400
P5	0.9218	0.9233	0.9226
P6	0.9103	0.9133	0.9118
P7	0.9422	0.9233	0.9327
P8	0.9532	0.9850	0.9689
P9	0.9635	0.9667	0.9651
P10	0.9589	0.9733	0.9661

Cuadro 4.4: Métricas de desempeño para cada nivel de severidad.

Class	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	575	6	3	1	5	1	1	3	0	5
P2	6	572	1	2	3	5	0	4	2	5
P3	2	3	557	5	14	10	2	3	2	2
P4	0	4	3	564	7	8	7	3	4	0
P5	3	2	11	5	554	11	6	1	3	4
P6	4	6	6	5	7	548	11	4	6	3
P7	3	0	1	12	3	14	554	6	4	3
P8	0	2	0	0	0	2	2	591	0	3
P9	0	5	0	6	4	1	3	1	580	0
P10	1	0	2	0	4	2	2	4	1	584

Cuadro 4.5: Matriz de confusión obtenida de la evaluación del modelo en el conjunto de prueba.

Código	MSE	MedAE
SCAE3 con capa de discretización	$7{,}985 \cdot 10^{-3}$	$5{,}332 \cdot 10^{-9}$
SCAE3 sin capa de discretización	$5{,}853 \cdot 10^{-3}$	$4{,}073 \cdot 10^{-5}$

Cuadro 4.6: Desempeño cuantitativo entre la inclusión y exclusión de la capa de discretización.

4.4.2. Incidencia de la capa de discretización

Esta prueba se realiza con una capa de discretización tras el regresor logístico y sin ella. Los resultados se muestran en la Tabla 4.6 y presentan un mejor desempeño general de SCAE sin la capa de discretización con un error medio cuadrático menor que el otro caso.

Además, los resultados indican que la probabilidad de salida del regresor logístico brinda información relacionada con la severidad del daño.

4.4.3. Comparación con otros métodos de extracción de características

Con el fin de comparar el método propuesto con otros se realizan dos bloques de comparativas. El primero usa métodos no-supervisados de extracción de características sin conocimiento de la tarea específica para la obtención de éstas. El segundo usa un método supervisado de extracción y selección de características junto con un clasificador como representante del enfoque clásico (ver Figura 4.13).

Para minimizar el sesgo en las comparativas causado por la selección aleatoria en el conjunto de entrenamiento y prueba, se realiza una validación cruzada de 5-folds, tanto para los métodos supervisados como para los no-supervisados.

Métodos no-supervisados de extracción de características

La comparación entre el método propuesto y otros 5 métodos se realiza con la exactitud como métrica. Los 5 métodos adicionales propuestos son:

1. T-PCA-MLP[71]: Aplica Análisis de Componentes Principales (PCA) a la señal de vibración. Para la selección del número de componentes principales se utiliza el criterio de Estimación de Máxima Verosimilitud de Minka, obteniendo de esta forma 765 características. Las características seleccionadas son las entradas para un perceptrón multi-capa con la misma estructura de su equivalente usado en SCAE3.

2. F-MLP[108]: Se aplica la Transformada Rápida de Fourier (FFT) a la señal de vibración. Todas las componentes espectrales son usadas como características

que ingresan a un perceptrón multicapa con la misma arquitectura que la usada en SCAE3.

3. F-PCA-MLP[71]: La FFT se aplica a la señal de vibración seguido por una fase de extracción de características por medio de PCA. Para la selección del número de componentes principales se utiliza el criterio de Estimación de Máxima Verosimilitud de Minka, obteniendo 780 características. Las características seleccionadas sirven de entrada para un perceptrón multicapa con la misma arquitectura que la usada en SCAE3.

4. TF-MLP: Se obtiene la representación tiempo-frecuencia de la señal de vibración usando el método propuesto en este trabajo. La matriz resultante es transformada a un vector de 8704 características apilando cada una de sus filas. Las características obtenidas son las entradas para un perceptrón multicapa con la misma arquitectura que la usada en SCAE3.

5. TF-PCA-MLP: Se obtiene la representación tiempo-frecuencia de la señal de vibración usando el método propuesto en este trabajo. La matriz resultante es transformada a un vector de 8704 elementos apilando cada una de sus filas. Se obtiene una vector de 754 características representativas usando PCA con el criterio de Minka. Las características obtenidas son las entradas para un perceptrón multi-capa con la misma arquitectura que la usada en SCAE3.

6. TF-SVM: Con el fin de comparar los resultados de nuestra propuesta con otro clasificador, se obtiene la representación tiempo-frecuencia de la señal de vibración. La matriz resultante es transformada a un vector de 8704 elementos apilando cada una de sus filas. El vector de 8704 características se aplica como entrada para una Máquina de Soporte Vectorial (SVM) multiclase de tipo *uno contra todos* con un kernel gaussiano.

Los resultados obtenidos son presentados en la Tabla 4.7, y muestran que los métodos tradicionales de extracción de características no supervisada no son capaces de extraer características informativas para la tarea presentada. Por una parte, esto se debe a la alta variabilidad en la señal de vibración con información redundante, pero principalmente a la combinación de señales estacionarias y no-estacionarias en los múltiples parámetros de operación, que oculta las características importantes. Se ha reportado en la literatura el uso de estas técnicas en condiciones de operación estacionarias o mínimamente no-estacionarias con buenos resultados, pero fallan en escenarios más complejos.

Métodos de extracción de características clásicos

Se realiza una comparación adicional con el uso del enfoque clásico para el diagnóstico de fallos basado en el aprendizaje automático. Las señales de vibración son procesadas para extraer las características estadísticas propuestas por [77]. Estas características pertenecen a los dominios de tiempo, frecuencia y tiempo-frecuencia,

Code	fold 1	fold 2	fold 3	fold 4	fold 5	avg. acc
SCAE3	95.02	95.05	95	94.22	95.53	$94{,}96 \pm 0{,}47$
T-PCA-MLP	24.1	26.22	17.46	20.56	13.78	$20{,}42 \pm 5$
F-MLP	10.08	10.5	10	10.71	10.42	$10{,}34 \pm 0{,}3$
F-PCA-MLP	17.67	20.01	18.52	15.05	16.45	$17{,}54 \pm 1{,}91$
TF-MLP	15.4	13.56	15.63	17.2	14.28	$15{,}21 \pm 1{,}39$
TF-PCA-MLP	24.3	26.12	25.47	22.5	26.23	$24{,}92 \pm 1{,}56$
TF-SVM	15.02	15.13	14.58	14.63	14.87	$14{,}85 \pm 0{,}24$

Cuadro 4.7: Comparación de SCAE con otros métodos no-supervisados de extracción de características. Cada columna "partición"tiene la exactitud evaluada en la partición correspondiente. La columna final tiene la exactitud promedio entre todos las particiones con su desviación estándar.

Code	fold 1	fold 2	fold 3	fold 4	fold 5	avg. acc
SCAE3	95.02	95.05	95	94.22	95.53	$94{,}96 \pm 0{,}47$
Rel+SVM	88	89.33	84	90.67	89.33	$88{,}27 \pm 2{,}56$

Cuadro 4.8: Comparación de SCAE con RELIEF+SVM. Cada columna "partición" tiene la exactitud evaluada en la partición correspondiente. La columna final tiene la exactitud promedio entre todos las particiones con su desviación estándar.

obteniendo un conjunto de 817 indicadores de la condición. A continuación se aplica un proceso de filtrado de indicadores correlacionados, lo que da 183 características no-correlacionadas que son normalizadas con media 0 y desviación estándar 1. Luego, se seleccionan 16 características usando RELIEF como técnica de selección de características, junto con un clasificador multiclase basado en una SVM de tipo uno contra todos con kernel gaussiano propuesto por [94]. El resultado es un sistema de clasificación del nivel de severidad de daño en el componente mecánico.

Los resultados se muestran en la Tabla 4.8, que muestra evidencias claras de un mejor desempeño en la estimación de la severidad del daño mediante el uso de características diseñadas por expertos en complemento con un proceso de selección de características y clasificación. Sin embargo en comparación con el método propuesto, RELIEF+SVM tiene una exactitud de $6{,}69\,\%$ por debajo de SCAE3. Además es posible notar que existe una pequeña variabilidad de $0{,}47\,\%$ en los resultados con SCAE3 frente a $2{,}56\,\%$ con RELIEF+SVM, lo que justifica que la propuesta de este trabajo es independiente de los datos usados para entrenar el modelo.

CAPÍTULO 5

APRENDIZAJE DE LA REPRESENTACIÓN Y ONE-CLASS LEARNING

5.1. Introducción

Las metodologías que hemos presentado en los capítulos anteriores nos han permitido alcanzar niveles de automatización cada vez más altos en el modelado de sistemas dinámicos a partir de información parcial adquirida de sus comportamientos. Las aplicaciones que pueden ser abordadas con estas metodologías cubren un amplio rango de problemas, que tienen como factor común la estimación del valor de una variable oculta del sistema dinámico a partir de la evolución temporal de variables visibles. Como ejemplos están las aplicaciones de CBM usadas como casos de estudio transversal a lo largo de toda esta memoria, en las cuales a partir de las series temporales obtenidas de señales medidas (variables visibles), se realizan inferencias sobre el estado de un componente interno de la máquina (variable oculta). Para este propósito, hasta este momento hemos utilizado instancias conocidas de pares $(serie\ de\ tiempo,\ variable\ oculta)$ que han permitido obtener por medio de conocimiento experto (Capítulo 3), o sin él (Capítulo 4), una representación de las series de tiempo para luego ser clasificadas en algún estado de la variable oculta.

A pesar de que las metodologías anteriores son muy útiles en multitud de aplicaciones, se encuentran limitadas por el elevado coste que, la mayoría de las veces, implica obtener pares $(serie\ de\ tiempo,\ variable\ oculta)$. Esto se debe a que se necesita un etiquetado de la variable oculta que normalmente deberá ser realizado por expertos en el área de aplicación, o que incluso puede resultar imposible de obtener. Siguiendo con el ejemplo tratado aquí, en una aplicación industrial de mantenimiento basado en la condición comúnmente se desea detectar que una pieza sufre algún daño o deterioro en etapas tempranas. Con esta información se pueden elabo-

rar planes de mantenimiento y/o el subsecuente reemplazo a tiempo del elemento. Sin embargo, tener información de múltiples niveles de daño de una gran cantidad de elementos (ejes, rodamientos, engranes, etc) resulta imposible por el tiempo requerido para alcanzar un cierto nivel de daño que permita realizar una inspección visual y categorizarlo. Para contextualizar lo dicho, un rodamiento de características estándar bajo condiciones de funcionamiento nominal puede tardar entre 4 y 5 años de funcionamiento continuo sin presentar fallas y, además, al aparecer podrían no ser fácilmente visibles, requiriendo detener la máquina para inspecciones continuas. Los costes de este procedimiento lo hacen totalmente inviable en situaciones prácticas donde además se pueden presentar daños en múltiples elementos al mismo tiempo. Bajo las circunstancias descritas, obtener un conjunto de datos etiquetados sobre las diferentes categorías resulta inviable.

Sin embargo, en múltiples circunstancias es económico disponer de datos etiquetados solamente para una categoría de la variable oculta. Retornando al ejemplo anterior, tras una revisión rutinaria de los componentes o de un cambio planificado de los mismos se sabe que estos se encuentran en buenas condiciones. Este conocimiento puede ser utilizado para capturar datos de series de tiempo bajo condiciones normales (etiquetadas de igual forma).

En este capítulo presentaremos una metodología para modelar un sistema dinámico únicamente a partir de series de tiempo pertenecientes a una sola categoría. A este tipo de aprendizaje se le conoce como *one-class learning* y ha sido usado en aplicaciones donde se tienen características explícitas extraídas a partir de conocimiento experto. Sin embargo, aquí se propone el uso de una técnica para el aprendizaje de la representación de una serie de tiempo basada en la Computación con Reservorios, que da como resultado un conjunto de vectores de estado que serán tratados como instancias muestreadas desde una variable aleatoria con distribución de probabilidad compleja. A partir de estas instancias se encuentra un modelo que aproxima su distribución de probabilidad y que, además, es capaz de generar nuevas instancias. Como resultado, este modelo entrega una métrica de similaridad entre nuevas instancias con la distribución de probabilidad aproximada, lo que permite determinar si la instancia pertenece a la clase aprendida o no.

Este capítulo se encuentra organizado como sigue. La Sección 5.2 proporciona el sustento teórico de las Redes Neuronales Recurrentes (RNN, por sus siglas en inglés) como mecanismo para el modelado de sistemas dinámicos. Aquí se abordan distintas arquitecturas, su formalización, posibles algoritmos de entrenamiento y las debilidades que presentan estos algoritmos para su uso generalizado por la dificultad que entraña entrenarlas. Una forma de superar los problemas de las RNN clásicas en el entrenamiento se presenta en la Sección 5.3 con el paradigma de computación con reservorios, donde se profundiza en el modelo de Echo State Network (ESN) con detalles acerca de su implementación, propiedades y algoritmos de entrenamiento. Posteriormente, en la Sección 5.4 se presentan los fundamentos teóricos de Variational Autoencoder (VAE), un modelo generativo que aúna las ideas de la inferencia variacional con Deep Learning para encontrar relaciones probabilísticas y complejas en un conjunto de datos. La Sección 5.5 introduce una metodología para el modelado

de sistemas dinámicos mediante one-class learning usando ESN para la extracción no-supervisada de características y VAE para el modelado probabilístico. La aplicación de la metodología anterior en un caso de estudio del mantenimiento basado en la condición se presenta en la Sección 5.6, donde el modelo resultante se utiliza para la detección de fallos en engranes helicoidales, rodamientos, y cajas de engranes rectos a partir de sus mediciones de vibración. Utilizando los resultados obtenidos en la detección de fallos en engranes helicoidales, en esta misma sección se realizan pruebas de estrés sobre los parámetros ESN, y finalmente se presenta una comparación con dos metodologías basadas en un conocido modelo para one-class learning.

5.2. Redes neuronales recurrentes

Como se puso de manifiesto en el Capítulo 4, las Redes Neuronales Artificiales (ANN) son un modelo de computación muy potente que es capaz de encontrar relaciones complejas entre las entradas y las salidas de la red. Estas relaciones son representadas como una composición de funciones, normalmente no-lineales, que son ajustadas por un conjunto de parámetros asociados a los pesos de la red. La composición expande las capacidades de representación funcional de la red estando solamente limitada por el número de capas y neuronas (elementos de cómputo) en cada una de las capas, y que deben ser elegidos de acuerdo a la complejidad del problema. Esta capacidad de cómputo ha llevado a demostrar que las ANN son aproximadores universales de cualquier función [38].

Además, la potencia de cómputo de las ANN puede ser incrementada mediante la inclusión de operaciones más complejas en sus neuronas. Bajo este precepto, históricamente el modelo lineal más simple de neurona evolucionó al modelo no-lineal sigmoidal, que presenta grandes ventajas en el proceso de entrenamiento. Posteriormente se han propuesto nuevas operaciones como núcleo de ANN más avanzadas, como las redes neuronales convolucionales (CNN, véase Capítulo 4), que usan la convolución como base de su cómputo, logrando que las neuronas se transformen en funciones de filtrado con capacidades de procesadores de señales. Sin embargo, independientemente de la potencia de cómputo que tenga cada tipo de ANN, el objetivo común es el mismo: obtener un modelo que aproxime una función capaz de mapear un conjunto de entradas, o variables independientes, a un conjunto de salidas, o variables dependientes.

Sin embargo, existen algunos inconvenientes al intentar modelar todos los procesos con un conjunto de variables predictoras independientes mapeadas a otras variables de salida. Por un lado, se encuentra la dificultad asociada a la existencia de un orden fundamental entre las variables predictoras, que no es abordada por las ANN clásicas[1] debido a la arquitectura típica de red completamente conectada que existe entre sus capas, por lo que impide un orden natural entre los datos de entrada. Otro

[1]Este problema sí es abordado por las CNN, que son capaces de encontrar patrones espaciales en la entrada de la red.

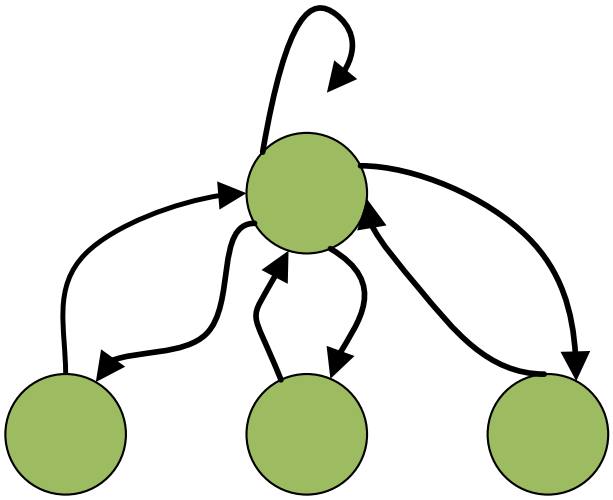


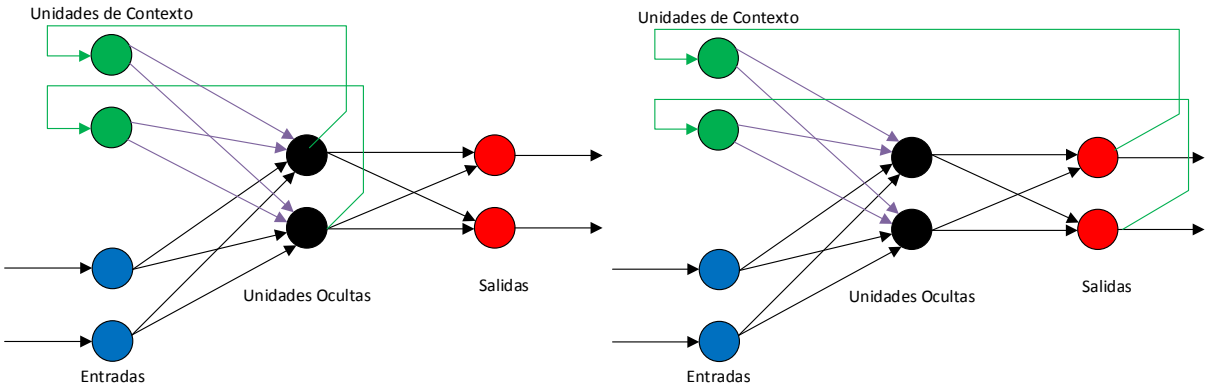
Figura 5.1: Representación de una RNN.

inconveniente aparece cuando las variables predictoras, y/o las salidas, son funciones del tiempo (continuo o discreto). En este caso, el mapeo entrada-salida, que en el modelo clásico solo se ve afectado por los valores instantáneos de la entrada a la red, ahora también es dependiente del momento en el que se realiza el mapeo. Finalmente, están los casos en donde para un paso de computación en la red se requiere el resultado de computaciones anteriores, ya sea de las salidas o de algún elemento interno. Las ANN clásicas, por su arquitectura feedforward, no son capaces de lidiar con esta dificultad al no disponer de mecanismos de retroalimentación, por ejemplo por medio de lazos o ciclos, que permitan reutilizar respuestas anteriores.

Por las razones citadas, y tomando como inspiración el funcionamiento de las redes neuronales del cerebro de los mamíferos, surgió un nuevo modelo de computación biológicamente más plausible denominado *Red Neuronal Recurrente* (RNN, por sus siglas en inglés), que permite la presencia de mecanismos de retroalimentación de estados anteriores para el cómputo de los nuevos estados de las neuronas (véase Figura 5.1), y que intenta replicar las conexiones que existen en un cerebro real, que en general no disponen de un orden jerárquico específico definido por un flujo de información unidireccional.

Los modelos iniciales de redes recurrentes fueron el modelo de Elman y el modelo de Jordan. Ambos añaden al modelo feedforward tradicional nuevos elementos de cómputo denominados *unidades de contexto* para el modelo de Elman, y, *unidades de la capa de estado* para el modelo de Jordan (ver Figura 5.2). En el primer caso, estas unidades almacenan el resultado del cómputo de las neuronas de la capa oculta y lo reinyectan como entradas para la misma capa de neuronas en la siguiente iteración. En cambio, en el segundo caso las unidades de la capa de estado almacenan el resultado de la salidas, que en la siguiente iteración son reinyectadas como entradas para el cómputo en la capa oculta.

Como se mencionó anteriormente, en la arquitectura de las RNN existen ciclos,



(a) Modelo de Elman (b) Modelo de Jordan

Figura 5.2: Primeros modelos de RNN.

Aspecto	ANN	RNN
Arquitectura	Propagación hacia adelante a través de capas ocultas.	Presencia de ciclos que permiten retropropagación.
Objeto matemático representado	Funciones.	Sistemas dinámicos.
Entrenamiento	Algoritmos basados en backpropagation.	Múltiples algoritmos propuestos.

Cuadro 5.1: Diferencias entre ANN clásicas y RNN.

lo que de forma natural permite el modelado explícito de relaciones temporales. Es por esta razón que han sido utilizadas de forma satisfactoria en tareas como la identificación de sistemas dinámicos, clasificación de patrones con relaciones temporales, modelado de series de tiempo deterministas o estocásticas, etc. Estas tareas aparecen en multitud de campos, tanto en problemas de ingeniería como de ciencia en general, dando paso a una gran diversidad de aplicaciones como son el procesamiento del lenguaje natural, control de sistemas industriales, filtrado de señales en las telecomunicaciones, etc.

En resumen, las diferencias entre las ANN clásicas y las RNN pueden ser caracterizadas por tres aspectos fundamentales: arquitectura, representación matemática, y entrenamiento. La Tabla 5.1 muestra una comparativa entre los dos modelos en cada uno de estos aspectos.

5.2.1. Descripción Formal de las RNN

Las RNN son modelos de computación que emulan de una manera muy simplificada la interacción entre los elementos de las redes neuronales biológicas, y en consecuencia su modelado es muy similar al de las redes neuronales clásicas. Esta interacción se da por la conexión con una determinada fuerza entre neuronas mediante sus enlaces sinápticos, que para propósitos de modelado es representada por el peso. En este modelo, al igual que en las ANN clásicas, se puede distinguir la presencia de 3 capas completamente identificadas: capa de entrada, oculta, y de salida, aunque veremos que presentan diferencias fundamentales con respecto a las capas equivalentes de las ANN. Para realizar una descripción formal del modelo, y como suele ser habitual, cada elemento se representará en notación matricial.

Para un paso de computación t[2], las entradas a la red reciben un vector $\mathbf{u}(t)$, de tamaño N_{in}, igual al número de neuronas presente en la capa de entrada:

$$\mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_{N_{in}}(t) \end{bmatrix} \tag{5.1}$$

Este vector de entradas es procesado en la capa oculta, generando a su vez un vector de activación $\mathbf{h}(t)$, de tamaño N_x, igual al número de neuronas presentes en la capa oculta:

$$\mathbf{h}(t) = \begin{bmatrix} h_1(t) \\ h_2(t) \\ \vdots \\ h_{N_x}(t) \end{bmatrix} \tag{5.2}$$

Este nuevo vector es igualmente procesado en la capa de salida para finalmente obtener un vector $\hat{\mathbf{y}}(t)$, de tamaño N_{out}, igual al número de neuronas presentes en la capa de salida:

$$\hat{\mathbf{y}}(t) = \begin{bmatrix} \hat{y}_1(t) \\ \hat{y}_2(t) \\ \vdots \\ \hat{y}_{N_{out}}(t) \end{bmatrix} \tag{5.3}$$

La Figura 5.3 muestra una representación gráfica de un modelo de RNN general.

Podemos representar las conexiones entre las diversas capas de la red haciendo uso de notación matricial. Así, las conexiones existentes entre la capa de entrada y

[2]En este contexto, t representa tiempo discreto, y para las RNN es equivalente al instante en el que se realizan todas las operaciones de computación dentro de la red.

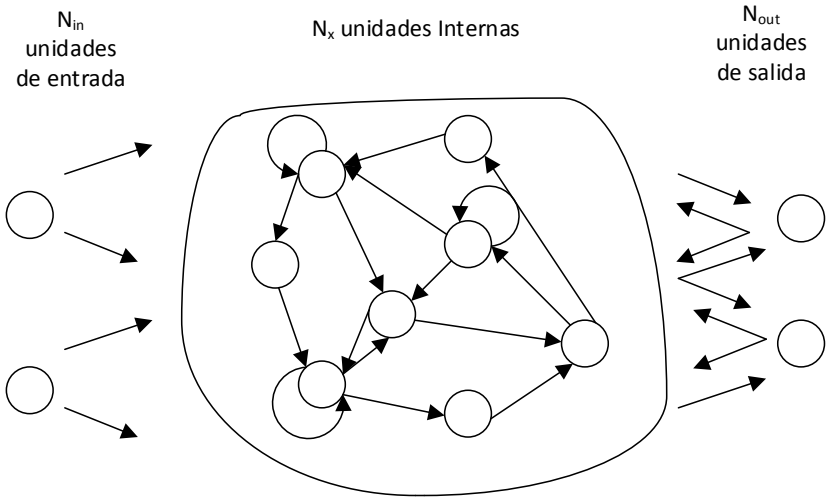


Figura 5.3: Modelo de Red Recurrente.

la capa oculta se presentan con la matriz W_{in} de tamaño $N_x \times N_{in}$:

$$W_{in} = \begin{bmatrix} w^{in}_{1,1} & w^{in}_{1,2} & \dots & w^{in}_{1,N_{in}} \\ w^{in}_{2,1} & w^{in}_{2,2} & \dots & w^{in}_{2,N_{in}} \\ \vdots & \vdots & \ddots & \vdots \\ w^{in}_{N_x,1} & w^{in}_{N_x,2} & \dots & w^{in}_{N_x,N_{in}} \end{bmatrix} \tag{5.4}$$

donde la columna i tiene los pesos de las conexiones existentes entre la i-ésima entrada y cada una de las neuronas de la capa oculta.

En las RNN la capa que normalmente contiene ciclos es la capa oculta. Esta interacción entre las neuronas de la misma capa está dada por la matriz de adyacencia W de tamaño $N_x \times N_x$:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,N_x} \\ w_{2,1} & w_{2,2} & \dots & w_{2,N_x} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N_x,1} & w_{N_x,2} & \dots & w_{N_x,N_x} \end{bmatrix} \tag{5.5}$$

donde la columna i tiene los pesos entre la i-ésima neurona de la capa oculta con el resto de neuronas de la misma capa.

Por último, la interacción entre la capa oculta y la de salida se da por la matriz W_{out} de tamaño $N_{out} \times N_x$:

$$W_{out} = \begin{bmatrix} w^{out}_{1,1} & w^{out}_{1,2} & \dots & w^{out}_{1,N_x} \\ w^{out}_{2,1} & w^{out}_{2,2} & \dots & w^{out}_{2,N_x} \\ \vdots & \vdots & \ddots & \vdots \\ w^{out}_{N_{out},1} & w^{out}_{N_{out},2} & \dots & w^{out}_{N_{out},N_x} \end{bmatrix} \tag{5.6}$$

donde la columna i tiene los pesos entre la i-ésima neurona de la capa oculta y cada una de las unidades de salida.

Por último, es posible que existan conexiones recurrentes entre la salida y la capa oculta (como en el modelo de Jordan). Esta interacción se modela con la matriz W_{fb} de tamaño $N_x \times N_{out}$:

$$W_{fb} = \begin{bmatrix} w^{fb}_{1,1} & w^{fb}_{1,2} & \dots & w^{fb}_{1,N_{out}} \\ w^{fb}_{2,1} & w^{fb}_{2,2} & \dots & w^{fb}_{2,N_{out}} \\ \vdots & \vdots & \ddots & \vdots \\ w^{fb}_{N_x,1} & w^{fb}_{N_x,2} & \dots & w^{fb}_{N_x,N_{out}} \end{bmatrix} \tag{5.7}$$

donde la columna i tiene los pesos entre la i-ésima neurona de salida y las diversas unidades de la capa oculta.

Esta interacción entre las distintas capas de la red depende del tipo de arquitectura recurrente utilizada. Por ejemplo, para el modelo de Elman la computación de los estados $\mathbf{h}(t)$ y la salida $\hat{\mathbf{y}}(t)$ viene dada por:

$$\mathbf{h}(t) = f\big(W_{in}\mathbf{u}(t) + W\mathbf{h}(t-1)\big) \tag{5.8}$$

$$\hat{\mathbf{y}}(t) = g\big(W_{out}\mathbf{h}(t)\big) \tag{5.9}$$

donde $f(\cdot)$ y $g(\cdot)$ son funciones normalmente no-lineales aplicadas elemento a elemento a su vector argumento.

En cambio, en el modelo de Jordan la computación de la red viene dada por:

$$\mathbf{h}(t) = f\big(W_{in}\mathbf{u}(t) + W_{fb}\hat{\mathbf{y}}(t-1)\big) \tag{5.10}$$

$$\hat{\mathbf{y}}(t) = g\big(W_{out}\mathbf{h}(t)\big) \tag{5.11}$$

Nótese que en el modelo de Elman la recurrencia viene dada por los ciclos creados entre las unidades de la capa oculta solamente, mientras que en el de Jordan los ciclos se crean por una conexión recurrente entre la respuesta de las salidas anteriores con el cálculo del estado actual de las neuronas de la capa oculta.

Un modelo más completo se obtiene fusionando los dos anteriores, obteniendo un modelo que considera recurrencias tanto entre las mismas unidades de la capa oculta como entre la salida y la capa oculta. De esta manera, la ecuación de activación de las unidades de la capa oculta es modificada con:

$$\mathbf{h}(t) = f\big(W_{in}\mathbf{u}(t) + W\mathbf{h}(t-1) + W_{fb}\hat{\mathbf{y}}(t-1)\big) \tag{5.12}$$

En algunas ocasiones resulta conveniente modificar la ecuación (5.11) para que el cálculo de la salida de la red contemple directamente las entradas, con lo que la ecuación resultante sería:

$$\hat{\mathbf{y}}(t) = g\big(W_{out}[\mathbf{h}(t); \mathbf{u}(t)]\big) \tag{5.13}$$

donde $[\mathbf{h}(t); \mathbf{u}(t)]$ es la concatenación vertical de los vectores $\mathbf{h}(t)$ y $\mathbf{u}(t)$. Como consecuencia de esta modificación, la matriz W_{out} cambiará de tener N_x columnas a $N_x + N_{in}$ columnas.

5.2.2. Entrenamiento de las RNN

El entrenamiento de las RNN puede seguir tanto un enfoque supervisado como no-supervisado. Para el enfoque supervisado se requiere un conjunto de datos de entrenamiento que conste de instancias formadas por secuencias de entrada relacionadas con secuencias de salida obtenidas del sistema dinámico que se desea modelar. Como es habitual en el entrenamiento de modelos de aprendizaje, el objetivo es encontrar los pesos óptimos en cada capa de la red para que ésta sea capaz de generar salidas similares al proceso que se está modelando ante las mismas entradas inyectadas (tanto a la red como al proceso).

Función de costo

Al igual que para las ANN clásicas, para abordar la tarea de entrenamiento es necesario primero definir una función de costo que permita cuantificar qué tan bien se encuentran las predicciones de la red con respecto a los datos reales. Para ello se presenta una secuencia de salida definida en un intervalo de tiempo $0, ..., T$, representada en forma del vector Y como:

$$Y = \begin{bmatrix} \mathbf{y}(0) \\ \mathbf{y}(1) \\ \vdots \\ \mathbf{y}(t) \\ \vdots \\ \mathbf{y}(T) \end{bmatrix} \tag{5.14}$$

y su correspondiente secuencia de predicción dada por la RNN como $\hat{Y}$:

$$\hat{Y} = \begin{bmatrix} \hat{\mathbf{y}}(0) \\ \hat{\mathbf{y}}(1) \\ \vdots \\ \hat{\mathbf{y}}(t) \\ \vdots \\ \hat{\mathbf{y}}(T) \end{bmatrix} \tag{5.15}$$

Cada elemento de Y (respectivamente, de $\hat{Y}$) representa el vector de salidas reales (respectivamente, de salidas predichas por la red) para cada instante de tiempo en la secuencia[3]. Como se había especificado anteriormente, el tamaño de cada uno de estos vectores es N_{out}.

En este contexto, una función de coste $E(Y, \hat{Y})$ es aquella que cuantifica el error existente entre la secuencias Y y $\hat{Y}$. Como se trata de encontrar el error entre secuencias, es posible representar el coste total E como la suma de las funciones de coste

[3]Por ninguna razón se debe considerar la agrupación de todos los $\mathbf{y}$ o $\hat{\mathbf{y}}$ en forma de vector como una operación de concatenación.

parciales $E_t(\mathbf{y}(t), \hat{\mathbf{y}}(t))$, para cada instante t en donde se encuentran definidas las secuencias:

$$E = \sum_{t=0}^{T} E_t \tag{5.16}$$

Regla de descenso del gradiente para RNN

A partir de aquí el problema se reduce a una tarea de optimización donde es necesario encontrar los parámetros adecuados de la red en el espacio de parámetros, $\theta = \{W_{in}, W, W_{out}, W_{fb}\}$, que minimicen la función de coste total, o lo que es igual, los parámetros $\hat{\theta}$ que minimicen las funciones de coste parciales, lo que se traduce en la siguiente expresión:

$$\hat{\theta} = \arg\min_{\theta} \sum_{t=0}^{T} E_t \tag{5.17}$$

De nuevo, una de las formas de resolver el problema de optimización anterior se logra mediante la regla de descenso del gradiente que, para una red recurrente con una función de activación de la capa oculta definida por la ecuación (5.12) y una activación en la capa de salida dada por la ecuación (5.11), permitiría actualizar cada parámetro de la red mediante:

$$W_{in} \leftarrow W_{in} - \eta \nabla_{W_{in}} E \tag{5.18}$$
$$W \leftarrow W - \eta \nabla_{W} E \tag{5.19}$$
$$W_{out} \leftarrow W_{out} - \eta \nabla_{W_{out}} E \tag{5.20}$$
$$W_{fb} \leftarrow W_{fb} - \eta \nabla_{W_{fb}} E \tag{5.21}$$

siendo $\nabla_{W_{in}} E$, $\nabla_{W} E$, $\nabla_{W_{out}} E$ y $\nabla_{W_{fb}} E$, las matrices $\frac{\partial E}{\partial W_{in}}$, $\frac{\partial E}{\partial W}$, $\frac{\partial E}{\partial W_{out}}$ y $\frac{\partial E}{\partial W_{fb}}$ con elementos i, j determinados por $\frac{\partial E}{\partial w_{i,j}^{in}}$, $\frac{\partial E}{\partial w_{i,j}}$, $\frac{\partial E}{\partial w_{i,j}^{out}}$ y $\frac{\partial E}{\partial w_{i,j}^{fb}}$ respectivamente. Por otro lado η representa la tasa de aprendizaje.

Back-Propagation Through the Time

En la situación actual el problema a resolver es el cómputo de las matrices anteriores. La aplicación directa del algoritmo de backpropagation no es posible debido a la presencia de ciclos en la red neuronal. Una modificación de este algoritmo fue propuesta por Paul Werbos en [102] con el nombre de *Backpropagation a través del tiempo* (BPTT por sus siglas en inglés, Backpropagation through the time). La idea fundamental de esta propuesta es expandir la RNN en múltiples redes pequeñas obtenidas a partir de una captura temporal de la red original, y que se interpretan como capas de una red más grande (ver Figura 5.4) con la finalidad de obtener una representación feedforward (sin ciclos) de la RNN. Es primordial notar que los

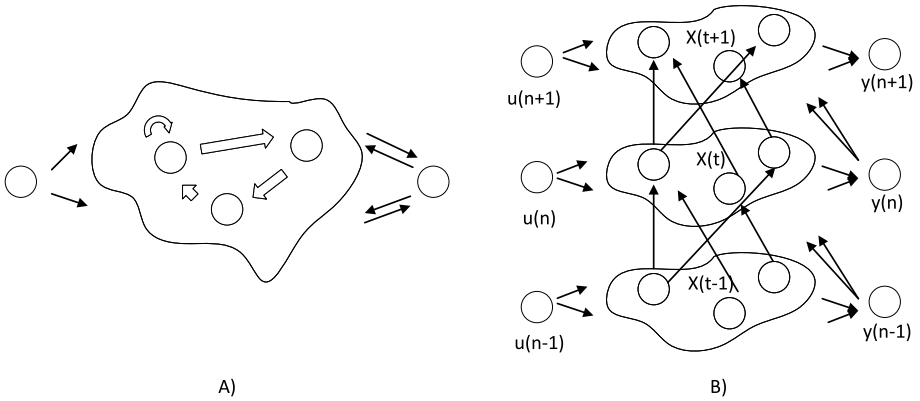


Figura 5.4: Entrenamiento con BPTT. a) RNN normal. b) Despliegue de la RNN.

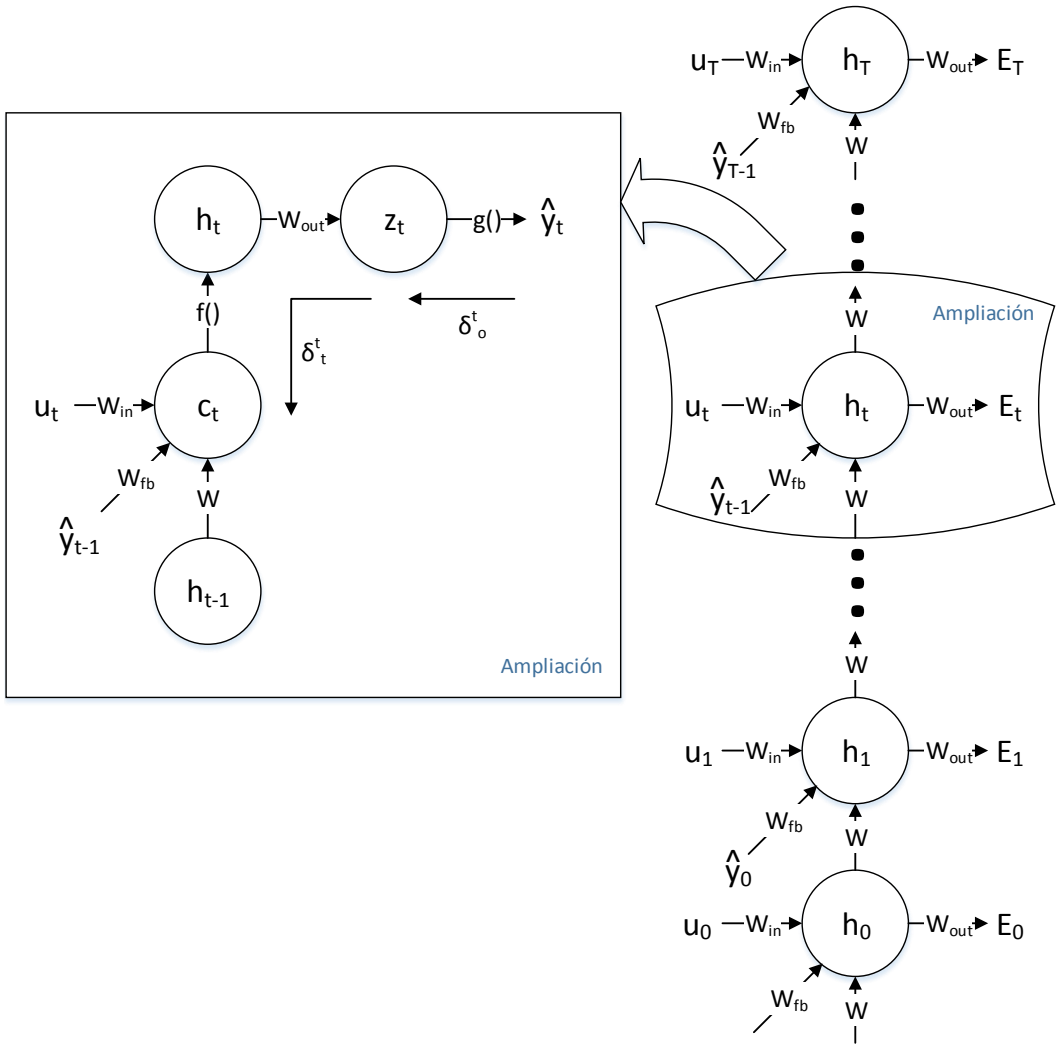


Figura 5.5: Representación simplificada de una RNN con ampliación en un capa.

parámetros de la RNN se replican en cada capa de la red expandida (por lo tanto se dice que la red resultante tiene parámetros compartidos).

Proponemos una representación simplificada de este procedimiento en la Figura 5.5, donde se expresan las dependencias existentes en la red expandida haciendo uso del tiempo como argumento (que se denota por medio de un subíndice)[4]. Además, en la representación propuesta se muestra una ampliación hacia los elementos que interactúan en la capa t, apareciendo dos nuevos componentes, llamados $\mathbf{c}_t$ y $\mathbf{z}_t$, para las operaciones lineales en el cálculo de la activación de las neuronas de la capa oculta y la capa de salida, respectivamente:

$$\mathbf{c}_t = W_{in}\mathbf{u}_t + W\mathbf{h}_{t-1} + W_{fb}\mathbf{y}_{t-1} \tag{5.22}$$

$$\mathbf{z}_t = W_{out}\mathbf{h}_t \tag{5.23}$$

Representando la activación de la capa oculta y la salida en función de los dos nuevos componentes se obtiene:

$$\mathbf{h}_t = f(\mathbf{c}_t) \tag{5.24}$$

$$\hat{\mathbf{y}}_t = g(\mathbf{z}_t) \tag{5.25}$$

Antes de calcular cada uno de los gradientes de la función de coste con respecto a los parámetros de la red es necesario determinar una regla que permita retropropagar la variación de E_t con respecto a las activaciones de cada una de las capas, lo que anteriormente habíamos denominado la regla de actualización de δ. Para la capa t, la razón de cambio de la función de coste E_t con respecto a $\mathbf{z}_t$, $\frac{\partial E_t}{\partial \mathbf{z}_t}$, permite cuantificar el crecimiento o decrecimiento de la función de coste al modificarse la salida en la capa t. Esta razón es nombrada como δ_o^t y su cálculo es:

$$\delta_o^t = \frac{\partial E_t}{\partial \mathbf{z}_t} = \frac{\partial E_t}{\partial \hat{\mathbf{y}}_t} \odot \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{z}_t} = \frac{\partial E_t}{\partial \hat{\mathbf{y}}_t} \odot \frac{\partial g(\mathbf{z}_t)}{\partial \mathbf{z}_t} \tag{5.26}$$

Por otro lado, nótese que la salida en la capa t depende de las activaciones de la capa oculta en t. Está relación permite calcular el cambio de la función de coste con respecto a cambios en la t-ésima capa oculta, que llamaremos δ_t^t, retropropagando el valor δ_o^t anteriormente calculado. Lo que se expresa formalmente como:

$$\delta_t^t = \frac{\partial E_t}{\partial \mathbf{c}_t} = \Big[\frac{\partial E_t}{\partial \mathbf{h}_t}\Big] \odot \frac{\partial \mathbf{h}_t}{\partial \mathbf{c}_t} = \Big[\Big(\frac{\partial \mathbf{z}_t}{\partial \mathbf{h}_t}\Big)^T \frac{\partial E_t}{\partial \mathbf{z}_t}\Big] \odot \frac{\partial f(\mathbf{c}_t)}{\partial \mathbf{c}_t} \tag{5.27}$$

$$= \Big[W_{out}^T \frac{\partial E_t}{\partial \mathbf{z}_t}\Big] \odot \frac{\partial f(\mathbf{c}_t)}{\partial \mathbf{c}_t} = W_{out}^T \delta_o^t \odot \frac{\partial f(\mathbf{c}_t)}{\partial \mathbf{c}_t} \tag{5.28}$$

Del mismo modo, las activaciones de la capa oculta en t dependen de las activaciones en la capa oculta en $t-1$, que a su vez dependen de las activaciones en $t-2$, y así sucesivamente. Esta relación recurrente continúa hasta la capa 0. Los cambios que

[4]En el cambio de notación de tiempo a capas, la entrada $\mathbf{u}(t)$ se reemplaza por la expresión $\mathbf{u}_t$. El mismo concepto se aplica para los términos restantes.

ocurren en las activaciones de cada capa oculta modifican E_t. Tomando como base la relación recurrente dada anteriormente es posible calcular cada razón de cambio de E_t con respecto a las activaciones de la capa oculta $k-1$ para k entre 1 y t. Esto se deduce con las siguientes operaciones:

$$\delta_{k-1}^t = \frac{\partial E_t}{\partial \mathbf{c}_{k-1}} = \frac{\partial E_t}{\partial \mathbf{c}_k} \odot \frac{\partial \mathbf{c}_k}{\partial \mathbf{c}_{k-1}} \tag{5.29}$$

$$= \delta_k^t \odot \frac{\partial}{\partial \mathbf{c}_{k-1}} \Big[W_{in}\mathbf{u}_k + W\mathbf{h}_{k-1} + W_{fb}\hat{\mathbf{y}}_{k-1} \Big] \tag{5.30}$$

Resolviendo cada término de la derivada parcial anterior se obtiene:

$$\frac{\partial W_{in}\mathbf{u}_k}{\partial \mathbf{c}_{k-1}} = \mathbf{0} \tag{5.31}$$

$$\frac{\partial W\mathbf{h}_{k-1}}{\partial \mathbf{c}_{k-1}} = W\frac{\partial f(\mathbf{c}_{k-1})}{\partial \mathbf{c}_{k-1}} \tag{5.32}$$

$$\frac{\partial W_{fb}\hat{\mathbf{y}}_{k-1}}{\partial \mathbf{c}_{k-1}} = W_{fb}\frac{\partial g(W_{out}f(\mathbf{c}_{k-1}))}{\partial \mathbf{c}_{k-1}} \tag{5.33}$$

$$z_{k-1} = W_{out}f(\mathbf{c}_{k-1}) \tag{5.34}$$

$$\frac{\partial W_{fb}\hat{\mathbf{y}}_{k-1}}{\partial \mathbf{c}_{k-1}} = W_{fb}\Big[\frac{\partial g(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}} \odot \frac{\partial W_{out}f(\mathbf{c}_{k-1})}{\partial \mathbf{c}_{k-1}}\Big] \tag{5.35}$$

$$= W_{fb}\Big[\frac{\partial g(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}} \odot W_{out}\frac{\partial f(\mathbf{c}_{k-1})}{\partial \mathbf{c}_{k-1}}\Big] \tag{5.36}$$

Por lo tanto, la regla de actualización de δ queda como:

$$\delta_{k-1}^t = \delta_k^t \odot \Big[W\frac{\partial f(\mathbf{c}_{k-1})}{\partial \mathbf{c}_{k-1}} + W_{fb}\Big[\frac{\partial g(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}} \odot W_{out}\frac{\partial f(\mathbf{c}_{k-1})}{\partial \mathbf{c}_{k-1}}\Big]\Big] \tag{5.37}$$

En función de este término se puede representar el cálculo de cada de uno de los gradientes de E_t con respecto a los parámetros de la red. Así, el cómputo de $\frac{\partial E_t}{\partial W_{out}}$ queda como:

$$\frac{\partial E_t}{\partial W_{out}} = \frac{\partial E_t}{\partial \mathbf{z}_t} \otimes \frac{\partial \mathbf{z}_t}{\partial W_{out}} = \delta_o^t \otimes \mathbf{h}_t \tag{5.38}$$

El cálculo de $\frac{\partial E_t}{\partial W}$ se deduce mediante:

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \mathbf{c}_k} \otimes \frac{\partial \mathbf{c}_k^+}{\partial W} = \sum_{k=0}^{t} \delta_k^t \otimes \mathbf{h}_{k-1} \tag{5.39}$$

donde $\frac{\partial \mathbf{c}_k^+}{\partial W}$ es la derivada instantánea de la activación en la k-ésima capa oculta con respecto a W[5]. Sin embargo, nótese que se agregan las contribuciones de todas las

[5]Esta notación representa la derivada parcial de $\mathbf{c}_k$ con respecto a W sin considerar la dependencia entre las activaciones en $\mathbf{h}_{k-1}$ y W.

capas debido a la dependencia existente entre las activaciones de las capas ocultas anteriores con W, por lo que no se puede considerar su derivada parcial con respecto a W como la derivada de una constante. Este mismo principio se aplica para los gradientes con respecto a W_{in} y W_{fb}.

Para el caso de W_{in} se tiene:

$$\frac{\partial E_t}{\partial W_{in}} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \mathbf{c}_k} \otimes \frac{\partial \mathbf{c}_k^+}{\partial W_{in}} = \sum_{k=0}^{t} \delta_k^t \otimes \mathbf{u}_k \tag{5.40}$$

y para W_{fb}:

$$\frac{\partial E_t}{\partial W_{fb}} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \mathbf{c}_k} \otimes \frac{\partial \mathbf{c}_k^+}{\partial W_{fb}} = \sum_{k=0}^{t} \delta_k^t \otimes \hat{\mathbf{y}}_{k-1} \tag{5.41}$$

Se puede notar que para todos los gradientes el término δ_k^t puede ser calculado con la ecuación (5.28) en conjunto con la ecuación (5.26) cuando $k = t$, y, con la ecuación (5.30) cuando $k < t$.

Finalmente, los gradientes totales pueden ser obtenidos mediante:

$$\frac{\partial E}{\partial W_{in}} = \sum_{t=0}^{T} \frac{\partial E_t}{\partial W_{in}} \tag{5.42}$$

$$\frac{\partial E}{\partial W} = \sum_{t=0}^{T} \frac{\partial E_t}{\partial W} \tag{5.43}$$

$$\frac{\partial E}{\partial W_{out}} = \sum_{t=0}^{T} \frac{\partial E_t}{\partial W_{out}} \tag{5.44}$$

$$\frac{\partial E}{\partial W_{bf}} = \sum_{t=0}^{T} \frac{\partial E_t}{\partial W_{bf}} \tag{5.45}$$

Usando las deducciones anteriores se puede obtener un algoritmo para el cálculo de los gradientes. Para ello, primero es necesario realizar la propagación hacia adelante en la RNN expandida, algo que se puede hacer con el Algoritmo 1, y posteriormente se calculan los gradientes con el Algoritmo 2.

Condiciones y un caso concreto

Del análisis anterior se deduce que para el uso de la regla de descenso del gradiente con el algoritmo de BPTT se requiere que tanto las funciones de coste parciales, E_t, como las funciones de activación, $f(\cdot)$ y $g(\cdot)$, han de ser derivables.

Una función de coste comúnmente utilizada para tareas de regresión con RNN se define como:

$$E_t = \frac{1}{2}(\mathbf{y}_t - \hat{\mathbf{y}}_t)^T(\mathbf{y}_t - \hat{\mathbf{y}}_t) \tag{5.46}$$

Algorithm 1 Algoritmo de propagación en una RNN

$\mathbf{h}_{-1} \leftarrow 0$
$\hat{\mathbf{y}}_{-1} \leftarrow 0$
for $t = 0$ **to** T **do**
  $\mathbf{h}_t \leftarrow f\big(W_{in}\mathbf{u}_t + W\mathbf{h}_{t-1} + W_{fb}\hat{\mathbf{y}}_{t-1}\big)$
  $\hat{\mathbf{y}}_t \leftarrow g\big(W_{out}\mathbf{h}_t\big)$
end for
return $\mathbf{h}$, $\hat{\mathbf{y}}$

Algorithm 2 Algoritmo de BPTT

$\frac{\partial E}{\partial W_{in}} \leftarrow 0$
$\frac{\partial E}{\partial W} \leftarrow 0$
$\frac{\partial E}{\partial W_{out}} \leftarrow 0$
$\frac{\partial E}{\partial W_{fb}} \leftarrow 0$
for $t = T$ **to** 0 **do**
  $\delta_o^t \leftarrow \frac{\partial E_t}{\partial \hat{\mathbf{y}}_t} \odot \frac{\partial g(\mathbf{z}_t)}{\partial \mathbf{z}_t}$
  $\frac{\partial E}{\partial W_{out}} \leftarrow \frac{\partial E}{\partial W_{out}} + \delta_o^t \otimes \mathbf{h}_t$
  $\delta^t \leftarrow W_{out}^T \delta_o^t \odot \frac{\partial f(\mathbf{c}_t)}{\partial \mathbf{c}_t}$
  for $k = t$ **to** 0 **do**
    $\frac{\partial E}{\partial W} \leftarrow \frac{\partial E}{\partial W} + \delta^t \otimes \mathbf{h}_{k-1}$
    $\frac{\partial E}{\partial W_{in}} \leftarrow \frac{\partial E}{\partial W_{in}} + \delta^t \otimes \mathbf{u}_k$
    $\frac{\partial E}{\partial W_{fb}} \leftarrow \frac{\partial E}{\partial W_{fb}} + \delta^t \otimes \hat{\mathbf{y}}_{k-1}$
    $\delta^t \leftarrow \delta^t \odot \left[W \frac{\partial f(\mathbf{c}_{k-1})}{\partial \mathbf{c}_{k-1}} + W_{fb} \left[\frac{\partial g(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}} \odot W_{out} \frac{\partial f(\mathbf{c}_{k-1})}{\partial \mathbf{c}_{k-1}} \right] \right]$
  end for
end for
return $\frac{\partial E}{\partial W_{in}}$, $\frac{\partial E}{\partial W}$, $\frac{\partial E}{\partial W_{out}}$, $\frac{\partial E}{\partial W_{fb}}$

Del mismo modo, si la entrada a la red se encuentra normalizada, una elección típica en cuanto a $f(\cdot)$ y $g(\cdot)$ es la $tanh$.

Bajo estas premisas, tanto la función de coste como las funciones de activación no-lineal son derivables en todo su dominio, por lo que es posible redefinir los términos que gobiernan el algoritmo de backpropagation. Así, el cálculo de δ_o^t queda expresado como:

$$\delta_o^t = \frac{\partial E_t}{\partial \mathbf{z}_t} = \frac{\partial E_t}{\partial \hat{\mathbf{y}}_t}\frac{\partial \hat{\mathbf{y}}_t}{\mathbf{z}_t} = \frac{1}{2}\frac{\partial}{\partial \hat{\mathbf{y}}_t}\Big[(\mathbf{y}_t - \hat{\mathbf{y}}_t)^T(\mathbf{y}_t - \hat{\mathbf{y}}_t)\Big]\frac{\partial}{\partial \mathbf{z}_t}\Big[tanh(\mathbf{z}_t)\Big] \tag{5.47}$$

$$= (\hat{\mathbf{y}}_t - \mathbf{y}_t)\Big[1 - tanh^2(\mathbf{z}_t)\Big] = (\hat{\mathbf{y}}_t - \mathbf{y}_t)(1 - \hat{\mathbf{y}}_t^2) \tag{5.48}$$

En consecuencia, se puede representar $\frac{\partial E_t}{\partial W_{out}}$ como:

$$\frac{\partial E_t}{\partial W_{out}} = \delta_o^t \otimes \mathbf{h}_t = (\hat{\mathbf{y}}_t - \mathbf{y}_t)(1 - \hat{\mathbf{y}}_t^2) \otimes \mathbf{h}_t \tag{5.49}$$

Tomando como base lo anterior, δ_t^t y la regla de actualización de δ pueden ser redefinidos como:

$$\delta_t^t = W_{out}^T\delta_o^t \odot \frac{\partial f(\mathbf{c}_t)}{\partial \mathbf{c}_t} = W_{out}^T\Big[(\hat{\mathbf{y}}_t - \mathbf{y}_t)(1 - \hat{\mathbf{y}}_t^2) \otimes \mathbf{h}_t\Big] \odot (1 - \mathbf{h}_t^2) \tag{5.50}$$

$$\delta_{k-1}^t = \delta_k^t \odot \Big[W\frac{\partial f(\mathbf{c}_{k-1})}{\partial \mathbf{c}_{k-1}} + W_{fb}\Big[\frac{\partial g(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}} \odot W_{out}\frac{\partial f(\mathbf{c}_{k-1})}{\partial \mathbf{c}_{k-1}}\Big]\Big] \tag{5.51}$$

$$= \delta_k^t \odot \Big[W(1 - \mathbf{h}_{t-1}^2) + W_{fb}\Big[(1 - \hat{\mathbf{y}}_{t-1}^2) \odot W_{out}(1 - \mathbf{h}_{t-1}^2)\Big]\Big] \tag{5.52}$$

Basado en los resultados anteriores, los gradientes con respecto a los parámetros de entrada, capa oculta y retroalimentación de la salida son calculados con sus ecuaciones originales.

Pérdida del gradiente en las RNN's

La pérdida del gradiente, como se vio en el Capítulo 4, es un problema que afecta en general a las redes neuronales profundas que usan el algoritmo de backpropagation para su entrenamiento. Por otro lado las RNN, en su versión más simple, cuentan solamente con 3 capas (entrada, oculta y salida), lo que podría significar que el problema de pérdida del gradiente no se presenta en ellas. Lamentablemente, como se analiza a continuación, esto no es cierto.

Como hemos visto en el apartado anterior, el aprendizaje de las RNN se basa en una versión modificada del algoritmo de backpropagation llamado Back Propagation Through Time, que expande la RNN en múltiples capas que están conectadas con una arquitectura feedforward con parámetros compartidos entre todas las capas. Posteriormente se calculan los valores δ de cada capa con los cuales se actualizan los parámetros de la red. Estos valores δ son el resultado de la retropropagación de los

valores δ de las siguientes capas (proceso que fue definido de forma recursiva por la ecuación (5.37)).

Siguiendo el desarrollo anterior, el δ_0 de la primera capa ($t = 0$) para la función de coste en la capa t es igual a una multiplicación ponderada sucesiva de $\delta_t, \delta_{t-1}, ..., \delta_1$. Además, se debe notar que por la función no-lineal seleccionada, acotada entre -1 y 1, cada δ será un vector acotado entre 0 y 1. Entonces, δ en la primera capa será proporcional a una multiplicación sucesiva de vectores con valores pequeños, posiblemente insignificantes. Nótese además que, con cada multiplicación, los valores para el δ de la capa anterior caen de forma exponencial. Esto finalmente produce que las entradas, estados de la capa oculta, y salidas de las capas iniciales, tengan un efecto mínimo o incluso nula para la actualización de los parámetros de la red.

Dentro de la perspectiva de las RNN el fenómeno anterior puede ser visto como una pérdida de memoria a largo plazo de la red, en donde los eventos con una cierta distancia temporal no son tomados en cuenta para el aprendizaje. Es por esta razón que el cómputo de valores δ muy distantes al tiempo t resulta innecesario y por lo tanto se han propuesto modificaciones de BPTT con versiones que no retropropagan el error más allá de una cierta cota. Sin embargo, esta medida no soluciona el problema de la pérdida del gradiente.

En la misma línea de buscar una mejora para el algoritmo de entrenamiento se han propuesto algoritmos basados en el *Filtro Extendido de Kalman*[92] (EKF, por sus siglas en inglés) con buenos resultados para ciertas tareas. Por otra parte, una opción muy popular disponible en la actualidad es el modelo de RNN llamado *Long Short Term Memory*[37] (LSTM), que propone una modificación al modelo neuronal simple usado en las RNN para que sea capaz de mantener en memoria eventos importantes que ocurrieron a una gran distancia, pero incrementa el coste computacional de la red junto con el tiempo de entrenamiento. Otra propuesta totalmente diferente a las anteriores para el mejoramiento de las prestaciones en una RNN es la Computación con Reservorios que será presentada a continuación.

5.3. Computación con Reservorios y Echo State Network

Como hemos podido ver en las secciones anteriores, las RNN son herramientas potentes que han permitido a las redes neuronales aproximarse al modelado de sistemas dinámicos, sin embargo, su corta capacidad de memoria y su dificultad en el entrenamiento han supuesto grandes limitaciones a la hora de utilizarlas con fines prácticos. Debido a esto, en el año 2001 aparecieron dos nuevos enfoques completamente diferentes hacia el entrenamiento y diseño de las RNN propuestas bajo el nombre de Liquid State Machines[62], por Wolfgang Maass, y bajo el nombre de Echo State Networks, por Helbert Jaeger [41]. Actualmente el conjunto de técnicas derivadas de los dos enfoques es referido conjuntamente como *Computación con Reservorios* (RC, por su nombre en inglés, Reservoir Computing).

En el año 2005, Schiller y Steil en [88] notaron que al aplicar el entrenamiento

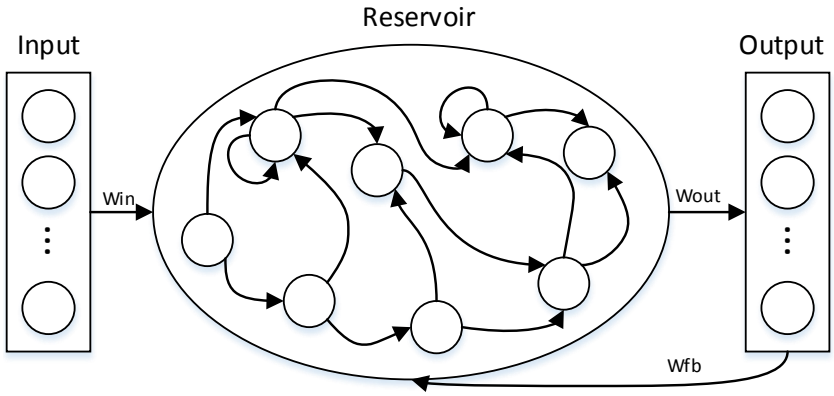


Figura 5.6: Arquitectura de una echo state network

con BPTT, los cambios más significativos ocurrían en los pesos de la capa de salida, mientras que los pesos en las capas más profundas casi no tenían cambio. Bajo dicha observación, la idea fundamental de RC es la de tratar por separado los pesos internos de la red y los pesos de la capa de salida. Mientras que en el aprendizaje supervisado tradicional el error entre la salida deseada y la salida calculada influye en toda la red, en el paradigma RC el error únicamente influye sobre los pesos de la capa de salida. A pesar de ser un concepto relativamente nuevo, la Computación con Reservorios ya tiene aplicaciones satisfactorias en el ámbito de la ingeniería y las ciencias.

Una de las variantes que se enmarcan en la RC son las Echo State Networks (ESN), que representan un modelo de red recurrente plausible desde un punto de vista biológico. Será esta variante la que tendremos en cuenta en el resto del capítulo.

5.3.1. Arquitectura

La Figura 5.6 muestra esquemáticamente la arquitectura de una ESN. Al igual que en las RNN genéricas, para un instante t la entrada a la red será representada con el vector $\mathbf{u}(t) \in \mathbb{R}^{N_{in}}$, la salida de la red con el vector $\hat{\mathbf{y}}(t) \in \mathbb{R}^{N_{out}}$, y la activación del conjunto de neuronas en el reservorio (equivalente a la capa oculta de las RNN) con $\mathbf{x}(t) \in \mathbb{R}^{N_x}$, también llamado *vector de estados*. Los parámetros de la capa de entrada, salida, reservorio, y retroalimentación de la red vienen dados por $W_{in} \in \mathbb{R}^{N_x \times N_{in}}$, $W_{out} \in \mathbb{R}^{N_{out} \times N_x}$, $W \in \mathbb{R}^{N_x \times N_x}$, y $W_{fb} \in \mathbb{R}^{N_x \times N_{out}}$, respectivamente.

5.3.2. Optimización entrada-reservorio

En este modelo los parámetros W_{in}, W y W_{fb} son elegidos y optimizados de forma independiente de los parámetros de la salida, prestando especial atención a la matriz W que define las conexiones internas del reservorio, y que deben cumplir con dos propiedades: *estado de eco* y *separabilidad*.

Que un reservorio verifique la propiedad de *estado de eco* significa que el efecto de la entrada y estados anteriores decae a medida que el tiempo transcurre, y de ninguna manera su efecto es mantenido, o peor aún, amplificado.

Si $\rho(W)$ denota el radio espectral de la matriz del reservorio (el valor absoluto del autovalor más grande de W), se puede comprobar que para el caso particular de $\mathbf{u}(t) = \mathbf{0}$ y función no-lineal $f = tanh$, para $\rho(W) > 1$ la propiedad de eco no se cumple y se dice que el reservorio es *inestable*. Por lo tanto, desde un punto de vista práctico se busca que $\rho(W) < 1$ aunque esto no garantice la estabilidad del reservorio. La elección del radio espectral está directamente relacionada con la cantidad de memoria que se requiere para la aplicación, de forma que a mayor $\rho(W)$ mayor capacidad de memoria del reservorio.

Por otra parte, que un reservorio verifique la propiedad de *separabilidad* quiere decir que es capaz de generar series de estados diferentes ante entradas diferentes. Es decir, si para t, que toma valores en $1...T$, una serie de tiempo de entrada $\mathbf{u_1}(t)$ genera una serie de estados $\mathbf{x_1}(t)$ y una entrada $\mathbf{u_2}(t)$ genera $\mathbf{x_2}(t)$ tal que $\mathbf{u_1}(t) \neq \mathbf{u_2}(t)$, entonces para que se cumpla la propiedad de separabilidad se debe cumplir que $\mathbf{x_1}(t) \neq \mathbf{x_2}(t)$.

Esta propiedad se puede conseguir por dos métodos fundamentalmente: 1) mediante la inicialización de W como una matriz dispersa desde un muestreo de una distribución normal estándar, y, 2) garantizando un número suficiente de neuronas en el reservorio.

El tamaño del reservorio está directamente ligado con la capacidad computacional de la red. Así, un reservorio lo suficientemente grande garantiza poder transformar el conjunto de estados en la señal $\mathbf{y}(t)$ solamente a partir de combinaciones lineales.

Los parámetros de entrada W_{in} normalmente se inicializan con un muestreo de la misma distribución que W, con la diferencia de que en este caso las conexiones son densas. En cambio, en la mayoría de aplicaciones, W_{fb} se inicializa con $\mathbf{0}$, a menos que se requiera una red capaz de generar una serie de tiempo sin haber recibido otra serie como entrada (es lo que se llama *modo free-running*).

5.3.3. Optimización de la periferia

Habiendo logrado un reservorio estable y con capacidades de cómputo y memoria suficientes, los únicos parámetros que faltan por optimizar están en la capa final de la red, los cuales a partir de un conjunto de estados conocidos y una serie de tiempo objetivo, $\mathbf{y}(t)$, representan los parámetros de un modelo de regresión lineal.

Como es habitual, la función de coste para este modelo viene dada por el error medio cuadrático (MSE) de la capa de salida:

$$E(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N_{out}} \frac{1}{T} \sum_{i=1}^{N_{out}} \sum_{t=1}^{T} [y_i(t) - \hat{y}_i(t)]^2 \tag{5.53}$$

donde $\hat{\mathbf{y}}(t)$ se calcula a partir de la matriz W_{out} y las activaciones del reservorio $\mathbf{h}(t)$. El valor óptimo de estos parámetros puede ser obtenido abordando el siguiente problema de optimización:

$$W_{out} = \arg\min_{W_{out}} E(\mathbf{y}, \hat{\mathbf{y}}) \tag{5.54}$$

Aunque es un problema de optimización similar a los anteriores, debido a que es una regresión lineal no es necesario utilizar métodos gradiente, y una forma sencilla de resolverlo es la siguiente: se procede a obtener un estado $\mathbf{x}(t)$ en cada instante de la señal de entrada $\mathbf{u}(t)$ (recordemos que t toma valores en el intervalo $1...T$); estos estados son apilados como columnas de una matriz que denotaremos por $\mathbf{X} \in \mathbb{R}^{N_x \times T}$. De igual forma, apilamos cada vector de salida deseado, $\mathbf{y}(t)$, en la matriz $\mathbf{Y} \in \mathbb{R}^{N_{out} \times T}$. Usando estas dos nuevas matrices, el problema de optimización de la ecuación 5.54 se puede resolver mediante Ridge Regression [67] como:

$$W_{out} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \beta I)^{-1} \tag{5.55}$$

donde I denota la matriz identidad.

5.4. Autoencoder variacional

El enfoque clásico del aprendizaje automático (supervisado) suele estar guiado por un procedimiento, más o menos estándar, para resolver cualquier problema de modelado basado en datos. En resumen, este enfoque clásico plantea la obtención de características de alto nivel de los datos por medio de conocimiento experto que permite construir un dataset compuesto por pares $(caracteristicas,\ valor)$, para luego construir un modelo por aprendizaje basado en datos que sea capaz de discriminar nuevas instancias. En consecuencia, un punto determinante en el éxito de la construcción de un modelo basado en datos es la calidad de las características obtenidas, que para propósitos del análisis de sistemas dinámicos representan variables de estado que determinan zonas específicas en el espacio de estados en donde aparecen órbitas que representan el estado actual del sistema en relación a alguna variable de interés. Si dichas características son de buena calidad las zonas en el espacio de estados para cada categoría de la variable de interés son fácilmente identificables y separables por modelos supervisados simples, e incluso pueden utilizarse modelos de aprendizaje no-supervisado para tareas de clustering.

Para alcanzar una buena calidad en las características se busca obtener independencia lineal y no-lineal entre ellas. Esto quiere decir que no exista una función que relacione las variables predictoras, o lo que es lo mismo, que no se pueda encontrar una transformación que permita llegar a unas desde otras. Por otra parte, es necesario que exista una alta dependencia entre las características y la variable de interés para que de esta forma sea más sencillo encontrar el modelo que las relaciona. La forma analítica más sencilla de conocer la independencia lineal entre variables

es mediante el coeficiente de correlación de Pearson, que establece una métrica de cuánta relación lineal existe entre dos variables. Sin embargo, medir la relación no-lineal entre dos variables resulta mucho más complejo.

Pero, ¿que sucede cuando las características no poseen independencia? Esta cuestión se ha abordado desde el aprendizaje automático con métodos de selección de características que consideran solamente aquellas que resulten más útiles. Sin embargo, existen problemas en donde los datos de entrada poseen una estructura intrínseca para los que, en primera instancia, no se puede/debe seleccionar un conjunto limitado de características. Por ejemplo, en una imagen donde se conozca que sus características determinantes son los bordes de un objeto resulta complicado seleccionar solamente ciertos pixels porque, incluso teniendo el conocimiento a priori de lo que se desea encontrar y su posible posición, en la mayoría de los casos existirán condicionantes (cambios en su posición por movimientos de la cámara, cambios del objeto, cambios en la luminosidad del ambiente) que impiden un correcto desempeño de una aproximación sencilla. Bajo estas circunstancias, es conveninete tomar en consideración todos los pixels (características), descartando cualquier suposición inicial de independencia y, de esta forma, flexibilizar el modelado ante datos de entrada estructurados.

Con este fin se presenta el *Autoencoder Variacional* (VAE, por su nombre en inglés, Variational Autoencoder) [25], un modelo de aprendizaje de tipo generativo con un fuerte enfoque probabilístico. Este modelo tiene como objetivo aprender distribuciones de probabilidad complejas que capturen la estructura de los datos provenientes de procesos no-lineales para posteriormente generar datos que sigan la distribución de probabilidad aprendida. Adicionalmente, proporciona una forma, menos costosa desde un punto de vista computacional, para determinar la verosimilitud de nuevos datos.

VAE es capaz de generar nuevas instancias de una distribución compleja aplicando primero un proceso de muestreo desde una distribución de probabilidad más simple, normalmente una Gaussiana, para posteriormente aplicar una transformación no-lineal diferenciable y parametrizada a los datos obtenidos del proceso anterior. Es decir, la transformación no-lineal puede surgir de un proceso de aprendizaje de parámetros basado en descenso del gradiente o similares. El proceso se justifica por el método de *Muestreo por Transformación Inversa* que se detalla a continuación.

5.4.1. Muestreo por Transformación Inversa

Supongamos que se desea generar datos de una variable aleatoria z que sigue una función de distribución acumulada (CDF) F_z. Para ello, vamos a suponer que se pueden muestrear datos de manera sencilla desde una variable aleatoria u que sigue una distribución de probabilidad uniforme U en el intervalo $[0, 1]$. Si aplicamos la función cuantil (la función inversa de una CDF) de F_z a la variable con distribución uniforme, aparece una nueva variable aleatoria que sigue la misma distribución de probabilidad, es decir $F_z^{-1}(u) = z$. En consecuencia, $F_z(z) = F_z(F_z^{-1}(u)) = u$), que

establece que F_z aplicado a z sigue una distribución uniforme en el intervalo $[0, 1]$.

De acuerdo con este hecho, se pueden generar muestras de una variable aleatoria x que sigue una función de distribución acumulada compleja, F_x, primero obteniendo una muestra desde una variable aleatoria continua arbitraria z y luego transformándola con:

$$x := F_x^{-1}(F_z(z)) \tag{5.56}$$

donde usualmente z sigue una distribución más simple que x.

Este resultado también es válido cuando tenemos $\mathbf{x}$ y $\mathbf{z}$ vectores representando a un conjunto de variables aleatorias. Sea cual sea el caso de análisis (uni o multidimensional), el problema de la formulación dado en la ecuación (5.56) radica en que la transformación dada por la composición de funciones $F_z \circ F_z^{-1}$ no se puede definir de forma explícita ni siquiera para una distribución simple como es la Gaussiana.

5.4.2. Autocodificación

La relación entre la variable (simple) $\mathbf{z}$ y la variable (compleja) $\mathbf{x}$ puede verse como si $\mathbf{z}$ codificara una estructura latente (oculta) encontrada en la variable observada $\mathbf{x}$. El caso determinista de este concepto se conoce como autocodificación (autoencoder, AE) que fue presentado con una aplicación concreta en la Sección 4.3. Para ese caso se aprende una representación razonable $\mathbf{z}$ mediante el aprendizaje de parámetros de dos redes neuronales simultáneamente, donde la primera red codifica $\mathbf{x}$ en $\mathbf{z}$, y la segunda red toma $\mathbf{z}$ y reconstruye una aproximación, $\hat{\mathbf{x}}$, de $\mathbf{x}$.

Para el caso del VAE se mantiene el mismo principio de autocodificación. Sin embargo, al hablar de variables aleatorias no se puede conservar un enfoque determinista para las fases de codificación y/o decodificación. Así pues, las principales diferencias entre el AE y el VAE son:

- La capa de codificación determinista del AE se reemplaza con unidades estocásticas.
- Para cada instancia obtenida de un proceso de muestreo de $\mathbf{x}$ se obtiene una distribución de probabilidad de codificaciones representados en $\mathbf{z}$ en la capa de codificación, en contra de la codificación única que, ante la misma entrada, se obtiene en el AE clásico.
- A partir de la distribución de codificaciones, $\mathbf{z}$, se calcula una distribución de probabilidad de $\mathbf{x}$ con una transformación no-lineal de $\mathbf{z}$, al contrario de la función no lineal que devuelve directamente la misma instancia $\mathbf{x}$ ante una misma codificación.
- La transformación no-lineal citada en el punto anterior es una aproximación de $F_x \circ F_z^{-1}$, y representa la solución propuesta por el método de muestreo por transformación inversa.

5.4.3. Formalización del modelo

Vamos a empezar el proceso de formalización a partir de nuestro objetivo inicial de modelar la verosimilitud $p(\mathbf{x})$ a la que se encuentra sujeta una variable aleatoria $\mathbf{x}$ que representa los datos de entrada al modelo. Como suponemos que esta distribución de probabilidad es muy compleja, introduciremos otra variable aleatoria $\mathbf{z}$ con distribución $p(\mathbf{z})$ más simple usando la función de probabilidad marginal de la siguiente forma:

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) dz \tag{5.57}$$

Considerando la otra probabilidad marginal, y mediante la regla de la multiplicación obtenemos que:

$$p(\mathbf{x}) = \int p(\mathbf{x}) p(\mathbf{z}|\mathbf{x}) dz \tag{5.58}$$

$$= p(\mathbf{x}) \underbrace{\int p(\mathbf{z}|\mathbf{x}) dz}_{1} \tag{5.59}$$

Continuando con el desarrollo de VAE vamos a aplicar una función monótona a la expresión como sigue:

$$ln\, p(\mathbf{x}) = ln\, p(\mathbf{x}) \int p(\mathbf{z}|\mathbf{x}) dz \tag{5.60}$$

La función aplicada es el logaritmo natural, ln, que al ser monótona creciente asegura que un proceso de optimización de $p(\mathbf{x})$ es igual a una optimización de la *logverosimilitud*, $ln\, p(\mathbf{x})$, es decir:

$$\arg\min p(\mathbf{x}) = \arg\min ln\, p(\mathbf{x}) \tag{5.61}$$

Otra razón importante del uso de la logverosimilitud es la estabilidad que brinda en la implementación gracias a las propiedades del logaritmo:

$$ln(f \cdot g) = ln(f) + ln(g) \tag{5.62}$$

$$ln(\frac{f}{g}) = ln(f) - ln(g) \tag{5.63}$$

ya que, desde un punto de vista práctico, en el cálculo computacional de funciones de probabilidad con precisión finita evitará errores por underflow que en otro caso ocurrirían por la multiplicación o división de números muy pequeños (entre 0 y 1 al ser distribuciones de probabilidad). Además, si las distribuciones usan exponenciales, como la función gaussiana que es la más usada, se evitará su cálculo con el logaritmo natural.

La distribución posterior $p(\mathbf{z}|\mathbf{x})$ de la ecuación (5.60) representa la distribución óptima de codificaciones $\mathbf{z}$ para la entrada $\mathbf{x}$, que suponemos que no se puede obtener analíticamente de forma sencilla. Vamos a intentar aproximar esta distribución

compleja por un modelo $q_\phi(\mathbf{z}|\mathbf{x})$ que está parametrizado con un conjunto de parámetros ϕ. En este caso, la logverosimilitud se reescribe como:

$$ln\, p(\mathbf{x}) = ln\, p(\mathbf{x}) \int q_\phi(\mathbf{z}|\mathbf{x}) dz \tag{5.64}$$

$$= \int q_\phi(\mathbf{z}|\mathbf{x}) ln\, p(\mathbf{x}) dz \tag{5.65}$$

De la regla de probabilidad condicional obtenemos que $p(\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{z}|\mathbf{x})}$, por lo que la expresión de logverosimilitud anterior se puede reescribir como:

$$ln\, p(\mathbf{x}) = \int q_\phi(\mathbf{z}|\mathbf{x}) ln\, \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} dz \tag{5.66}$$

$$= \int q_\phi(\mathbf{z}|\mathbf{x}) ln\, \frac{p(\mathbf{x},\mathbf{z}) q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x})} dz \tag{5.67}$$

En el paso anterior se incluye la aproximación de la posterior q_ϕ en el término del lado derecho que contiene al logaritmo. Aplicando las propiedades del logaritmo y separando los resultados en dos integrales queda:

$$ln\ p(\mathbf{x}) = \underbrace{\int q_\phi(\mathbf{z}|\mathbf{x}) ln\, \frac{p(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} dz}_{\mathcal{L}_{VAE}(q_\phi(\mathbf{z}|\mathbf{x}))} + \underbrace{\int q_\phi(\mathbf{z}|\mathbf{x}) ln\, \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} dz}_{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))} \tag{5.68}$$

Pasemos a analizar el segundo término de la última ecuación, marcado por D_{KL}, que denota la *divergencia Kullback-Leibler*, y que de forma general, para sus versiones discretas y continuas, viene dada por:

$$D_{KL}(P||Q) = \sum_{x \in X} p(x)\, ln \frac{p(x)}{q(x)} \tag{5.69}$$

$$D_{KL}(P||Q) = \int p(x)\, ln \frac{p(x)}{q(x)}\, dx \tag{5.70}$$

La KL divergencia, como es comúnmente llamada, se utiliza como métrica entre las distribuciones de probabilidad $p(x)$ y $q(x)$, donde el primero suele representar un modelo exacto y el segundo una aproximación. En este contexto, y de forma intuitiva, podemos decir que cuando $D_{KL}(P||Q)$ se aproxima a cero significa que la aproximación Q se acerca al modelo real P. Por otro lado, formalmente la KL divergencia no es una métrica al no cumplir la desigualdad triangular[6] ni es simétrica[7], aunque sí cumple la condición de no-negatividad[8].

[6]Desigualdad Triangular: $d(A,B) + d(B,C) \geq d(A,C)$.

[7]Simétrica: $d(A,B) = d(B,A)$.

[8]No-negativa: $d(A,B) \geq 0\, \forall A, B$.

De lo anterior podemos decir que $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$ representa la cercanía de la aproximación de la posterior $q_\phi(\mathbf{z}|\mathbf{x})$ al modelo exacto $p(\mathbf{z}|\mathbf{x})$). En consecuencia, minimizar la KL divergencia anterior permitiría obtener la mejor aproximación posible de la siguiente forma:

$$q_\phi = \arg\min_{q_\phi} D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) \tag{5.71}$$

Lamentablemente, este problema de optimización no es computable debido a que no se conoce el modelo exacto de la posterior, y de hecho esa fue la razón para introducir el modelo de aproximación de la posterior que estamos desarrollando.

A pesar de ello, la ecuación (5.68) puede todavía brindarnos información útil sobre cómo abordar el problema de modelado. Para un modelo con parámetros fijados, ϕ, la logverosimilitud de los datos, $ln\, p(\mathbf{x})$, permanece constante. Por lo tanto, un proceso de minimización de la KL divergencia para un espacio de modelos q_ϕ como el mostrado en la ecuación (5.71) es equivalente a un proceso de maximización en el mismo espacio de modelos del término marcado como $\mathcal{L}_{VAE}(q_\phi(\mathbf{z}|\mathbf{x}))$. Es decir:

$$\arg\max_{q_\phi} \mathcal{L}_{VAE}(q_\phi(\mathbf{z}|\mathbf{x})) = \arg\min_{q_\phi} D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) \tag{5.72}$$

Por ello, y sabiendo que la KL-divergencia es una medida no-negativa, resulta fácil observar que para una logverosimilitud de los datos constante lograda con un modelo q_ϕ, el término $\mathcal{L}_{VAE}$ es una cota inferior de la logverosimilitud, es decir, este término podrá, como mucho, llegar a ser igual a la logverosimilitud de los datos cuando la KL divergencia sea igual a 0, lo cual solamente se cumple cuando la aproximación de la posterior es exactamente el modelo buscado. Esto es:

$$\mathcal{L}_{VAE}(q_\phi(\mathbf{z}|\mathbf{x})) \leq ln\, p(\mathbf{x}) \tag{5.73}$$

Al ser la KL divergencia intratable y $\mathcal{L}_{VAE}$ solamente una cota inferior de $ln\, p(\mathbf{x})$ descartamos el hecho de poder optimizar de forma completa la ecuación (5.68). Sin embargo, aunque el resultado obtenido no sea el más óptimo por despreciar la KL divergencia, es posible continuar nuestro análisis solamente con la cota inferior.

Aplicando de nuevo la regla de la multiplicación a la distribución de probabilidad conjunta entre $\mathbf{x}$ y $\mathbf{z}$ se obtiene que:

$$\mathcal{L}_{VAE}(q_\phi(\mathbf{z}|\mathbf{x})) = \int q_\phi(\mathbf{z}|\mathbf{x}) ln\, \frac{p(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} dz \tag{5.74}$$

$$= \int q_\phi(\mathbf{z}|\mathbf{x}) ln\, \frac{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} dz \tag{5.75}$$

lo que permite separar la mencionada expresión en la distribución prior de la variable de codificación $p(\mathbf{z})$ y $p(\mathbf{x}|\mathbf{z})$ como la distribución de un modelo generativo del cual se pueden obtener instancias de $\mathbf{x}$ a partir de la variable $\mathbf{z}$.

Desarrollando la ecuación (5.75) usando las propiedades de los logaritmos obtenemos:

$$\mathcal{L}_{VAE}(q_\phi(\mathbf{z}|\mathbf{x})) = \int q_\phi(\mathbf{z}|\mathbf{x}) ln\, p(\mathbf{x}|\mathbf{z}) dz - \int q_\phi(\mathbf{z}|\mathbf{x}) \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} dz \tag{5.76}$$

donde, ahora, $\int q_\phi(\mathbf{z}|\mathbf{x}) \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} dz$ es la KL divergencia entre el modelo q_ϕ y la prior $p(\mathbf{z})$. Como nuestro objetivo es que la cota inferior sea máxima para que ésta sea lo más cercana posible a la logverosimilitud, entonces debemos minimizar esta nueva KL divergencia.

Este problema de optimización es abordable debido a que se trata de encontrar un modelo de codificación q_ϕ que, dado los datos, proporcione una codificación $\mathbf{z}$ que siga una distribución lo más cercana posible a $p(\mathbf{z})$, que puede ser elegida de forma predeterminada. A modo de ejemplo, si se elige que $p(\mathbf{z})$ siga una distribución normal estándar, es decir, con $\mu = 0$ y $\sigma = 1$ para cada elemento del vector $\mathbf{z}$, entonces se trata de encontrar un modelo q_ϕ, o más específicamente un conjunto de parámetros ϕ, que transforme los datos de entrada a una distribución con $\mu = 0$ y $\sigma = 1$.

Por otro lado, el término $\int q_\phi(\mathbf{z}|\mathbf{x}) ln\, p(\mathbf{x}|\mathbf{z}) dz$ deberá ser maximizado con el mismo fin. Para un q_ϕ fijo, este objetivo se puede lograr encontrando un modelo generativo p_θ con parámetros θ que reemplazará a $p(\mathbf{x}|\mathbf{z})$, el cual a partir de una muestra de la variable $\mathbf{z}$ que sigue una distribución $p(\mathbf{z})$ es capaz de obtener instancias de $\mathbf{x}$. Este modelo generativo es optimizado reduciendo el error de reconstrucción al generar nuevas instancias de $\mathbf{x}$.

De esta manera, hemos logrado obtener una forma clara de maximizar el término $\mathcal{L}_{VAE}$, que es un objetivo más realista. Por otro lado, hay que considerar que solamente estamos optimizando una cota inferior de la logverosimilitud, que no es lo más óptimo desde un punto de vista teórico pero es tratable desde un punto de vista computacional.

5.5. Metodología para un eficiente one-class learning en sistemas dinámicos

En la mayoría de casos, la única fuente de información disponible sobre un sistema dinámico proviene de series de tiempo que se extraen de mediciones de algunas de sus variables. Esta información es utilizada con múltiples fines, como por ejemplo estimar el estado de otras variables, predecir eventos futuros en el sistema dinámico, o detectar anomalías en su evolución.

Un enfoque orientado a datos para las tareas citadas anteriormente está basado en la caracterización del sistema dinámico por medición directa de la evolución de las variables de interés en sus diferentes estados, lo cual permite obtener un conjunto de tuplas de tipo $(serie\ de\ tiempo,\ etiqueta\ de\ estado)$, donde $etiqueta\ de\ estado$ es

el estado actual, estado futuro, o condición del sistema dinámico. Con estos datos se buscan patrones de alto nivel en las series de tiempo que permitan su mapeo hacia la etiqueta.

En consecuencia, una tarea de máximo interés para la aplicación de este enfoque es la captura de los datos en todos los estados del sistema dinámico, lo que permitiría encontrar los patrones que los representan. Como ya hemos comentado, esta tarea resulta excesivamente costosa (o imposible) desde la disponibilidad de los datos, debido a que no siempre es posible etiquetar todas las condiciones del sistema dinámico, ya sea por una limitada accesibilidad a la variable de interés o por el poco control que se tiene para modificar sus condiciones. En estos casos es necesario que el proceso de modelado basado en datos sea lo menos dependiente posible de la información proveniente de las etiquetas y ponga un mayor énfasis en patrones intrínsecos de las series de tiempo.

Propones aquí una metodología para el modelado de un sistema dinámico del que únicamente se conocen series de tiempo obtenidas en un estado concreto de la variable de interés, que se mantiene durante el proceso de obtención de los datos. Esta propuesta es robusta ante posibles cambios que el sistema dinámico pueda tener en el resto de variables.

La figura 5.7 muestra la metodología, compuesta por cuatro etapas principales:

1. Adquisición de las series de tiempo y preprocesamiento.
2. Extracción no-supervisada de características mediante el aprendizaje de la representación.
3. Aprendizaje de un modelo probabilístico sobre el nuevo espacio de representación.
4. Inferencia sobre el modelo.

Hagamos un recorrido más detallado sobre cada una de estas etapas.

5.5.1. Adquisición de las series de tiempo y preprocesamiento

Una señal medida en un sistema dinámico multi-componente puede exhibir un comportamiento caótico debido a la interacción entre los elementos que lo componen y también a causa de fuentes de ruido internas o externas [100]. Cuando existe periodicidad, con el propósito de mitigar los efectos del ruido y resaltar la información importante, las señales son normalmente medidas y promediadas usando sensores externos de referencia que sirvan para encontrar el inicio y el fin de cada periodo. Desafortunadamente, incluir dispositivos adicionales incrementa los costos y no siempre es físicamente posible realizar estas mediciones.

El método propuesto en este capítulo evita, en muchos casos, el uso de dispositivos de medición adicionales para la etapa de adquisición. Solamente requiere un simple paso de preprocesamiento.

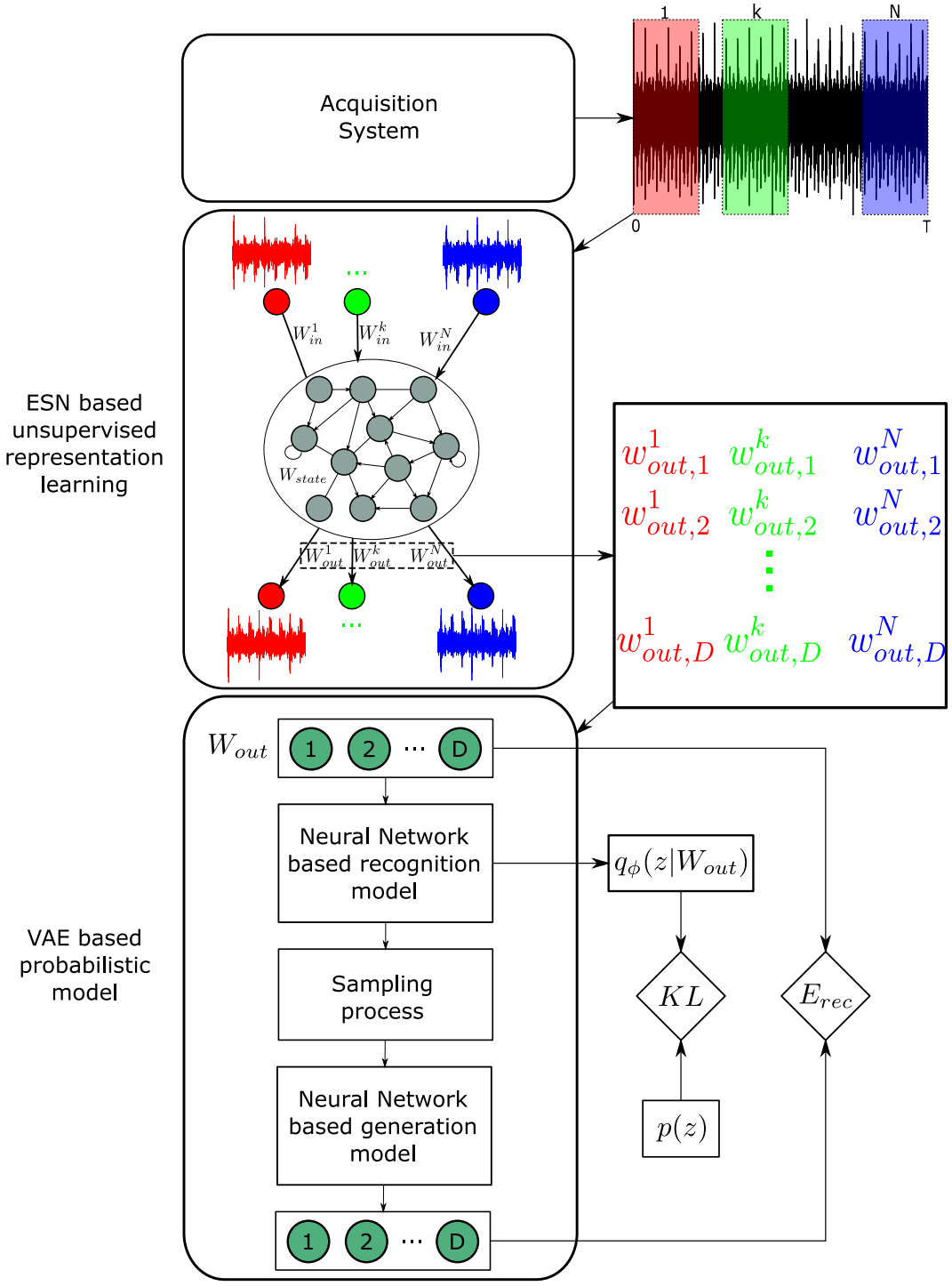


Figura 5.7: Método propuesto para la construcción del modelo ESN-VAE.

Si $\mathbf{y}_m(t)$ con $t = 1, ..., T$ es la señal medida, entonces se aplica un proceso de normalización estándar:

$$\mathbf{y}(t) = \frac{\mathbf{y}_m(t) - min(\mathbf{y}_m)}{max(\mathbf{y}_m) - min(\mathbf{y}_m)} \in [0, 1] \tag{5.77}$$

En lo que sigue consideraremos que las señales han sido normalizadas por este procedimiento.

Posteriormente, para cada señal disponible, $\mathbf{y}$, se extraen un conjunto de N subseñales ($\mathbf{y}^1, ..., \mathbf{y}^k, ..., \mathbf{y}^N$) usando una ventana deslizante de tamaño y paso prefijados. Está claro que, dependiendo de los parámetros de esta ventana, podría existir solapamiento entre las subseñales $\mathbf{y}^{k-1}, \mathbf{y}^k, \mathbf{y}^{k+1}$, lo cual puede ser deseable para capturar las relaciones temporales intrínsecas entre ellas.

5.5.2. Extracción no-supervisada de características

En esta etapa se obtiene un modelo predictivo para cada $\mathbf{y}^k$ usando una ESN para anticipar un paso de la serie de tiempo. Para este objetivo, el modelo ESN resultante puede ser formalizado como:

$$\mathbf{x}(t) = f[W_{in}\mathbf{y}(t-1) + W\mathbf{x}(t-1)] \tag{5.78}$$
$$\hat{\mathbf{y}}(t) = W_{out}\mathbf{x}(t) \tag{5.79}$$

donde se usa $\mathbf{y}(t-1)$ para calcular $\hat{\mathbf{y}}(t)$.

El numero de entradas N_{in} depende del número de señales obtenidas en las mediciones del sistema dinámico. Como el objetivo es predecir la misma señal de entrada, el número de salidas N_{out} es igual a N_{in}. Los parámetros W_{in} y W_{state} se inicializan de acuerdo a lo expuesto en la sección 5.3 para cada señal $\mathbf{y}^k$, es decir, el reservorio se vuelve a inicializar para cada subseñal. De esta forma, los únicos parámetros modificables durante el entrenamiento para predecir $\mathbf{y}^k$ son los elementos de W^k_{out}.

Usando esta ESN se desea que la predicción $\hat{\mathbf{y}}(t)$ sea cercana a la señal $\mathbf{y}(t)$. Esto indicaría que los parámetros W^k_{out}, vistos como los parámetros del modelo con una función base prefijada por el reservorio, son óptimos para reconstruir las señales medidas (normalizadas). Desde otra perspectiva, para una inicialización de W_{in} y W, W^k_{out} codifica una representación adecuada de la serie de tiempo de entrada. Si el tamaño de $\mathbf{y}^k$ es suficiente grande, esta codificación es invariante ante nuevos valores temporales en la entrada, lo que se puede interpretar como que la matriz resultante, $\mathbf{W_{out}}$, compuesta por las matrices W^k_{out} concatenadas por columnas, representa la serie de tiempo original, $\mathbf{y}$, en un nuevo espacio de representación concreto.

5.5.3. Aprendizaje de un modelo probabilístico

Teniendo en cuenta que el reservorio se inicializa de forma diferente para cada serie de tiempo, desde una perspectiva probabilística, $\mathbf{W_{out}}$ puede ser vista como una variable aleatoria con una distribución de probabilidad desconocida y compleja, que es difícil de modelar. Por ello, se propone el uso de un VAE que permitirá codificar la variable aleatoria $\mathbf{W_{out}}$ en una variable latente z más simple.

En VAE, tanto q_ϕ como p_θ pueden ser obtenidos con cualquier modelo de aprendizaje basado en datos. Sin embargo, el más utilizado en la mayoría de casos son las redes neuronales debido a que se sabe que son buenos aproximadores universales [38]. En consecuencia, haremos uso de una red neuronal para q_ϕ en la que, en el caso más común, sus salidas codifiquen una distribución de probabilidad gaussiana multivariable; y otra red para p_θ que funcionará como un modelo de reconstrucción desde la variable z hasta x. A estas redes las llamaremos, respectivamente, *red de codificación* y *red de generación*. Al ser parte de un mismo proceso de optimización de la cota inferior $\mathcal{L}_{VAE}$, las redes son entrenadas de forma conjunta (usando, por ejemplo, cualquier versión del algoritmo del gradiente descendente). Para realizar este entrenamiento conjunto se retropropaga el error de reconstrucción a través de las capas de la red de generación hasta la variable latente z, y luego se continúa con el mismo proceso desde z por las capas de la red de codificación.

En general, este proceso implica que se necesita retropropagar el error a través de una variable aleatoria, lo cual no es posible. Sin embargo, desde un punto de vista práctico, al elegir z con una distribución gaussiana, se puede representar como $z = u + \sigma \odot \epsilon$, con $\epsilon \sim \mathcal{N}(0, I)$ representando una entrada más a la red, lo que se conoce como el *truco de reparametrización*, que evita que se tenga que retropropagar el error por una variable aleatoria.

Así pues, para obtener los mejores parámetros del modelo VAE con dos redes neuronales se elige z como una variable que sigue una distribución normal estándar multi-variable. Por otro lado, se sabe que la pérdida por KL-divergencia y el error de reconstrucción constituyen la función de coste total a optimizar y, al haber impuesto $z \sim \mathcal{N}(\mu, cov)$ (donde cov es la matriz de covarianzas), éstos pueden ser obtenidos con las siguientes ecuaciones:

$$KL_{loss} = -\frac{1}{2N}\sum_{k=1}^{N}\sum_{j=1}^{|z|} 1 + ln[(\sigma_j^k)^2] - (\mu_j^k)^2 - (\sigma_j^k)^2 \tag{5.80}$$

$$E_{rec} = -\frac{1}{N}\sum_{k=1}^{N}\sum_{i=1}^{N_x} w_{out,i}^k \cdot ln(\hat{w}_{out,i}^k) + (1 - w_{out,i}^k) \cdot ln(1 - \hat{w}_{out,i}^k) \tag{5.81}$$

$$Loss = KL_{loss} + E_{rec} \tag{5.82}$$

5.5.4. Inferencia

Una vez construido el modelo, lo cual implica la ejecución de todas las etapas anteriores, se pueden realizar inferencias sobre el estado de la variable de interés a partir de nuevas series de tiempo adquiridas del sistema dinámico. Para ello, se almacenan W_{in} y W_{out} de la ESN, y también los parámetros de las redes neuronales de VAE.

Para evaluar una nueva señal se pasa por el mismo proceso de adquisición y preprocesamiento para generar un batch de series de tiempo. Luego se obtiene de cada elemento del batch una representación estática con la ESN, que en conjunto forman la matriz W_{out} de ese modelo. Posteriormente, cada k columna de W_{out} pasa por las redes de codificación y generación del modelo VAE en donde se calcula E^k_{rec}. A continuación, se calcula L_{avg}, que constituye una métrica que indica la cercanía entre la nueva señal con una señal correspondiente al estado del sistema dinámico a partir del cual se generó el modelo VAE. Para ello, L_{avg} se calcula por medio de :

$$L_{avg} = \frac{1}{N}\sum_{k=1}^{N} E^k_{rec} \tag{5.83}$$

Este valor nos indicará cómo de cerca o lejos estará la señal que representa la dinámica actual del sistema de las condiciones iniciales con las que fue entrenado el modelo.

5.6. Aplicación a la detección de fallas

La aplicación del método anterior a la detección de fallas en maquinaria rotativa trabaja en dos etapas:

1. Aprendizaje del modelo en estado normal.
2. Pruebas en línea.

El aprendizaje del modelo en estado normal es llevado a cabo minimizando la pérdida (ecuación 5.82) para un conjunto de señales de vibración adquiridas de diferentes condiciones operativas de velocidad y carga rotativas, pero todas ellas bajo la suposición de que el estado es normal (es decir, no hay fallas).

Por otro lado, la prueba en línea se realiza usando como entrada del modelo aprendido la señal de vibración desconocida, y tomando como salida el error de reconstrucción dado por la ecuación 5.83. En estas condiciones, valores negativos grandes (en valor absoluto) indican una alta probabilidad de que la señal de entrada se corresponde con aquellas utilizadas para entrenar el modelo aprendido, y por lo tanto es una medida de maquinaria sin falla. Por el contrario, si el valor es positivo, o negativo con valor absoluto pequeño, entonces hay una baja probabilidad de que

la señal sea generada usando el modelo aprendido, y por lo tanto es una medida de maquinaria con falla.

5.6.1. Configuración experimental

Para evaluar la metodología con diversos sistemas dinámicos se propone su aplicación en tres tareas distintas de mantenimiento basado en la condición:

- Detección de fallas en engranes helicoidales.
- Detección de fallas en rodamientos.
- Detección de fallas en cajas de engranes rectos.

Los pormenores de cada implementación son detallados a continuación.

Detección de fallas en engranes helicoidales

El primer conjunto de experimentos se llevó a cabo para identificar la presencia de falla de rotura de dientes en un engrane helicoidal a diferentes niveles de gravedad independientemente de la carga y velocidad de operación en el sistema mecánico.

Las pruebas se ejecutaron con una velocidad de eje de entrada de $480rpm$, $720rpm$ y $900rpm$ codificadas como F1, F2 y F3 respectivamente, con la configuración mostrada en la Figura 2.23. Para cada velocidad de entrada, tres cargas de salida de $0V$, $10V$, y $30V$ codificadas como L1, L2 y L3 respectivamente, son aplicados a través de un freno magnético controlado por una fuente de alta tensión de corriente y acoplado al eje de salida a través de una correa. El engrane de prueba es el piñón de entrada, y se realizaron diferentes niveles de rotura en uno de sus dientes, tal y como viene especificado en la Tabla 2.2. El piñón tiene un diámetro de $76mm$, 30 dientes, un ángulo de presión de $20°$ y un ángulo de hélice de $20°$.

En el estado normal del engrane, para cada posible combinación de velocidad y carga, fueron adquiridas 5 señales de vibración de 280001 muestras cada una (alrededor de 5.6seg), a 50k muestras por segundo, dando un conjunto de 45 señales de entrenamiento usadas para construir el modelo con el método propuesto. Se ha considerado una ventana deslizante con longitud de 50k muestras (1seg) y un paso deslizante de 10k muestras. Para la etapa de extracción de características no-supervisada, el tamaño de W^k_{out} es 1001, compuesto por 1000 pesos y 1 término de bias. El modelo de reconocimiento y el modelo de generación se componen cada uno por 2 capas ocultas de 1000 neuronas, y el tamaño del espacio latente z es de 20.

Para la prueba del modelo, para cada posible combinación de velocidad, carga, y estado (incluyendo el estado normal), se han adquirido 5 señales de vibración con las mismas condiciones anteriores, dando un total de 450 señales de prueba.

Detección de fallas en rodamientos

En este segundo conjunto de experimentos el objetivo es evaluar la metodología en la detección de fallas en los distintos elementos de un rodamiento (pista interna, pista externa y elemento rodante). La falla puede presentarse a distintos niveles de severidad en cualquiera de los elementos como lo muestran la Tabla 2.3, la Tabla 2.4 y la Tabla 2.5, pero el modelo debe ser capaz de detectarla independientemente del nivel de daño. Para este fin, y tomando en cuenta las restricciones que se presentan en los casos reales de adquisición de datos, únicamente se utilizan señales de vibración en condiciones normales para entrenar el modelo.

Al igual que en la tarea anterior, la evaluación bajo diferentes condiciones de operación, en la configuración mostrada en la Figura 2.25, es realizada mediante la adquisición de señales a 3 diferentes valores de velocidad y 3 cargas con la misma codificación vista anteriormente. Así, F1 representa una velocidad de $8Hz$, F2 una de $12Hz$ y F3 una de $15Hz$. Para la carga se tiene que L1 representa la presencia de rueda volante, L2 sin rueda volante pero con conexión al freno magnético por banda sin resistencia adicional, y L3 conexión al freno magnético con alimentación de $10V$.

Para el entrenamiento del modelo, sin la presencia de fallas, fueron adquiridas 5 señales de vibración para cada condición de operación de velocidad y carga, con un tamaño de 2000 muestras cada una (alrededor de 0.04seg), a 50k muestras por segundo, dando un conjunto de 45 señales de entrenamiento usadas para construir el modelo con el método propuesto. Se ha considerado una ventana deslizante con longitud de 1000 muestras (0.02seg) y un paso deslizante de 200 muestras. Para la etapa de extracción de características no-supervisada, el tamaño de W_{out}^k es 101, compuesto por 100 pesos y 1 término de bias. El modelo de reconocimiento y el modelo de generación se componen cada uno por 2 capas ocultas de 500 neuronas, y el tamaño del espacio latente z es de 20.

Para la prueba del modelo, para cada posible combinación de velocidad, carga, y estado representativo de fallo, se han adquirido 10 señales de vibración con las mismas condiciones anteriores. De igual forma se adquieren 5 señales por cada condición de operación para el estado normal, dando un total de 1485 señales de prueba.

Detección de fallas en cajas de engranes rectos

Finalmente, se ha realizado un conjunto de experimentos para evaluar la metodología para la detección de fallas en cajas de engranes rectos. Aquí el objetivo es determinar el desempeño del modelo resultante para una tarea en la que los estados son muy diversos entre sí, donde cada uno representa fallos en distintos componentes del sistema mecánico. La descripción de estos estados se muestra en la Tabla 2.1. Al igual que en los dos casos anteriores, la metodología se enfoca en la construcción de un modelo que permita discriminar el estado normal del resto de estados independientemente de las diferentes condiciones de velocidad y carga.

En el mismo sentido, para la etapa de entrenamiento solamente se disponen de señales de vibración en condición normal, de las cuales se capturan 45 de ellas para la construcción del modelo. La longitud de cada señal, ventana y paso deslizante son los mismos que en el caso anterior, así como el tamaño de W_{out}^{k}, el número de capas ocultas del VAE, y el tamaño del espacio latente z.

5.6.2. Resultados y Análisis

La Figura 5.8 muestra los resultados de evaluar el modelo aprendido para la detección de fallos en engranes helicoidales, con nuevas señales de vibración adquiridas. En ella se observa que el máximo valor de L_{avg} para el estado normal es de $-147{,}2$ dado en las condiciones (F1, L2) como se muestra en la Figura 5.8a. Por otro lado, con la presencia de falla el menor valor de L_{avg} es $-8{,}6$ en las condiciones (F2, L3) como muestra la Figura 5.8b, con una gran distancia intermedia de $138{,}4$ que incrementa el grado de confianza del resultado. El aumento de velocidad permite distinguir más notablemente la ausencia de falla con un valor máximo de $-391{,}2$ como se muestra en la Figura 5.8c.

Al analizar la Figura 5.8a, la Figura 5.8b y la Figura 5.8c conjuntamente se observa que en cargas más altas la ausencia de falla es más notable. En todos los casos los resultados se agrupan con la misma condición operativa de velocidad y carga.

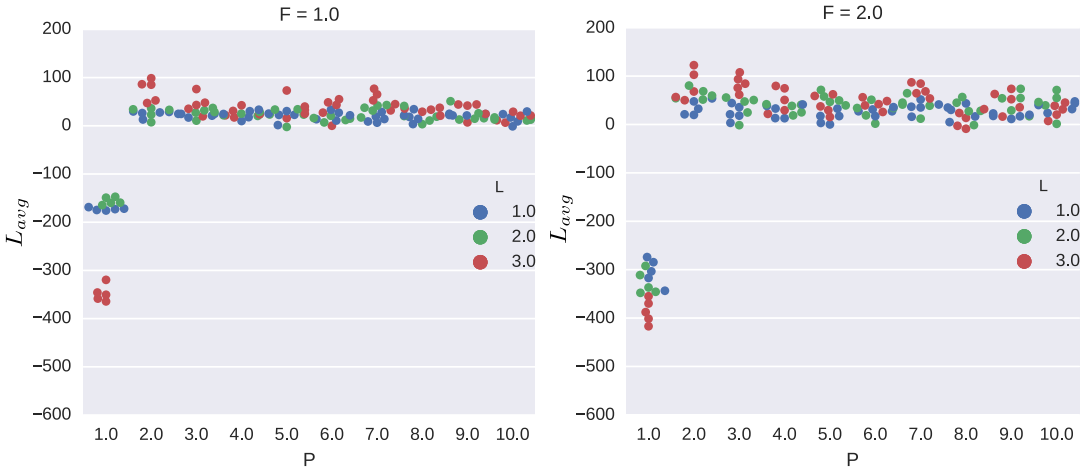
Independientemente de las condiciones de velocidad y carga, la detección de fallas incipientes como P2 o daños de severidad de alto nivel como P3-P7 no muestran dificultad en ser identificadas por el modelo aprendido. Debido a la distancia en el espacio L_{avg} entre el estado normal y el estado defectuoso es posible seleccionar un umbral de decisión haciendo uso de un amplio rango $(-8{,}6, -147{,}2)$, lo que permite obtener un $100\,\%$ de precisión en la tarea de detección de fallos.

La Figura 5.9 y la Figura 5.10 muestran los resultados para la detección de fallos en rodamientos y en cajas de engranes rectos, respectivamente. En el primer caso, se encuentra una diferencia de $96{,}74$ entre las instancias más cercanas del estado normal y la condición con falla. De forma similar, en el segundo caso la distancia es de $83{,}53$. Así, en las dos tarea se obtiene el $100\,\%$ de tasa de detección con un amplio margen para la selección del umbral.

En la tarea de detección de fallos en rodamientos se puede notar que el margen entre el estado normal y el resto de estados se incrementa tanto con la carga como con la velocidad, lo cual indicaría que este estado es más detectable en el espacio L_{avg} bajo condiciones de estrés del mecanismo. Por el contrario, en la caja de engranes rectos las variaciones a causa de la velocidad y carga son imperceptibles.

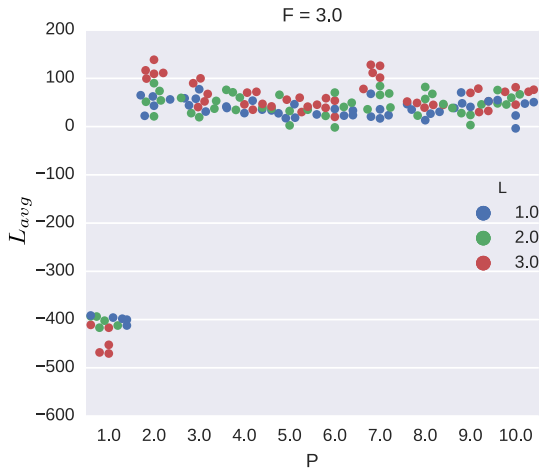
5.6.3. Pruebas de estrés

Con el fin de probar el desempeño ante variaciones en el tamaño de la señal de entrada a la ESN y tamaño del reservorio se propone la evaluación del modelo ge-



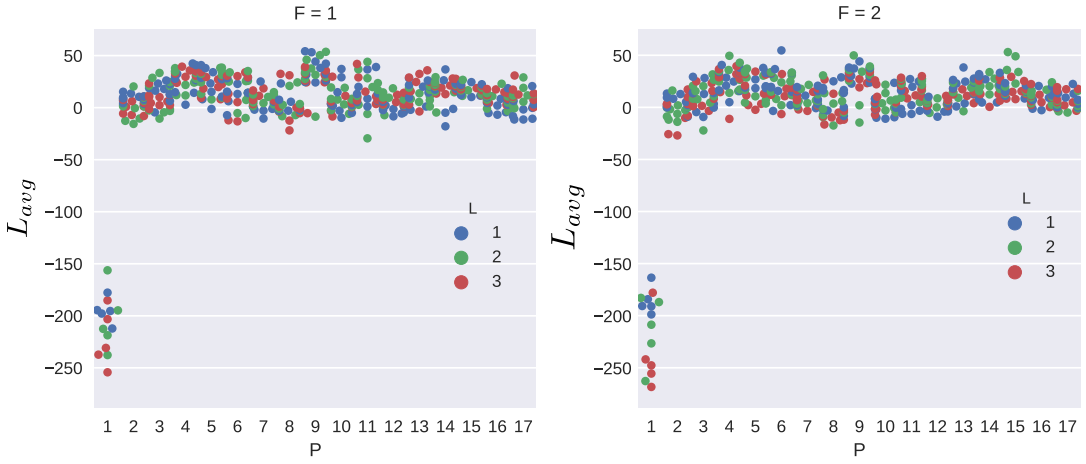
(a) Prueba para la velocidad 1

(b) Prueba para la velocidad 2



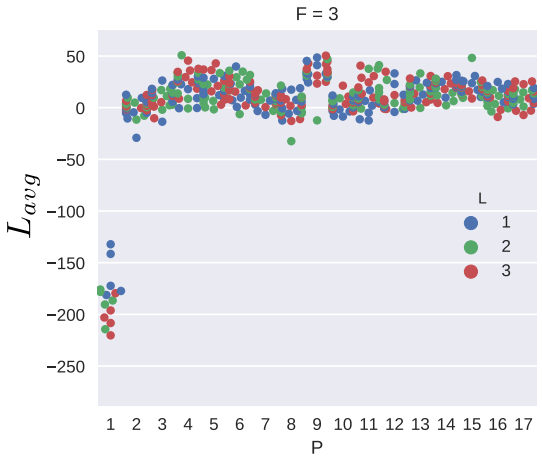
(c) Prueba para la velocidad 3

Figura 5.8: Resultados de las pruebas del modelo aprendido con todas las combinaciones de P-estados, F-velocidades y L-cargas en engranes helicoidales.



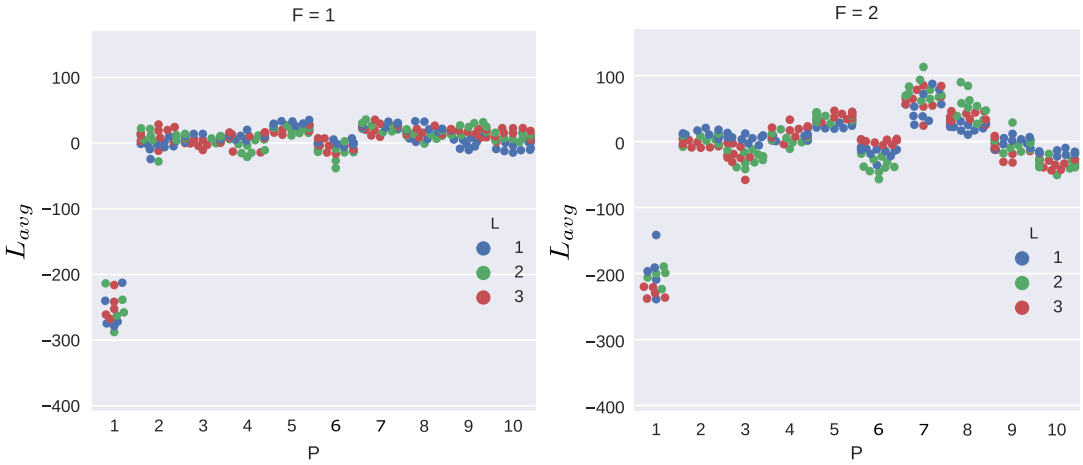
(a) Prueba para la velocidad 1

(b) Prueba para la velocidad 2



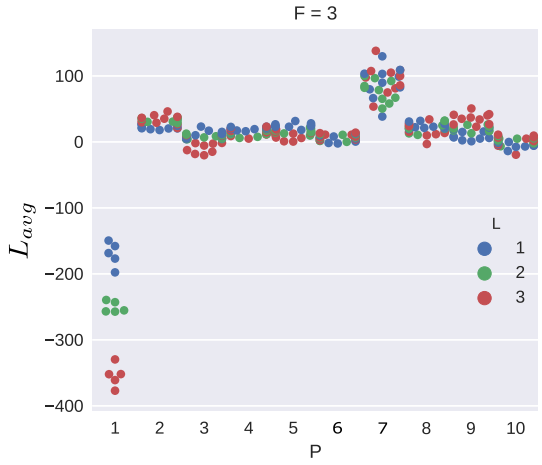
(c) Prueba para la velocidad 3

Figura 5.9: Resultados de las pruebas del modelo aprendido con todas las combinaciones de P-estados, F-velocidades y L-cargas para rodamientos.



(a) Prueba para la velocidad 1

(b) Prueba para la velocidad 2



(c) Prueba para la velocidad 3

Figura 5.10: Resultados de las pruebas del modelo aprendido con todas las combinaciones de P-estados, F-velocidades y L-cargas para engranes rectos.

neur./vent.	1000	200	150	120	100	50	20	10
100	74.94	22.3	11.58	4.57	-2.22	-6.85	-6.17	-5.55
50	69.98	15.23	-4.12	-9.81	-13.18	-13.53	-11.00	-12.99
20	30.53	5.75	-10.87	-3.31	-0.36	-14.06	-25.36	-11.59
10	14.77	-18.58	-6.78	-36.30	-4.88	-17.31	-28.86	-17.39

Cuadro 5.2: Pruebas de estrés para diferentes tamaños de ventana y número de neuronas en el reservorio.

nerado para diferentes valores de ventana deslizante y número de neuronas en el reservorio. La tarea seleccionada para esta prueba es la detección de fallos en engranes helicoidales.

Los valores seleccionados para el tamaño del reservorio son: 100, 50, 20, 10. Para cada uno de los reservorios anteriores se evalúa un tamaño de ventana deslizante de 1000, 200, 150, 120, 100, 50, 20, 10. La métrica usada nuevamente es el margen entre las instancias más cercanas del estado normal y las condiciones con falla en el espacio L_{avg}.

Los resultados de esta evaluación se muestran en la Tabla 5.2, donde se puede apreciar que, de forma general, si el modelo es capaz de discriminar de forma satisfactoria todas las instancias del estado normal de los estados con falla, el margen discriminante se incrementa con un mayor tamaño de la ventana deslizante. Por otro lado, el número de neuronas en el reservorio influye de forma inversa con el umbral mínimo necesario del tamaño de la ventana deslizante para obtener una correcta detección de todas las instancias. Esto es, a mayor número de neuronas, menor tamaño necesario para la ventana. Por ejemplo, con 100 neuronas en el reservorio el umbral de la ventana se ha de encontrar en un valor entre 100 y 120, en cambio con 50 neuronas el umbral cambia a un valor entre 150 y 200. Además, por debajo del umbral de detección correcta no existe una disminución lógica del margen con respecto a la disminución del tamaño de la ventana.

5.6.4. Estudio comparado

Con el fin de evaluar los modelos resultantes de la aplicación de la metodología propuesta frente a otras técnicas clásicas de One-Class Learning se propone la realización de dos comparativas.

La primera se enfoca en considerar los vectores obtenidos de los pesos en la capa de salida de la ESN como características[9] que representan a la subseñal, las cuales se usan para el aprendizaje de un modelo clásico para One-Class Learning basado

[9]Nótese que para el caso de VAE este vector se considera una instancia obtenida por muestreo de una variable aleatoria multidimensional.

en SVM y propuesto en el trabajo de Scholkopf [89]. Estos dos modelos en conjunto (ESN y SVM) se han denominado aquí como ESN-SVM.

La segunda elimina la etapa del aprendizaje no-supervisado de la representación realizado con la ESN y considera cada valor de la serie de tiempo en cada subseñal como una característica que la distingue. De igual forma, se usan estas características para el aprendizaje de un modelo clásico basado en SVM para la detección de un estado. Al modelo resultante se lo denomina TS-SVM.

En ambos casos la comparación se realiza para la tarea de detección de fallos en engranes helicoidales.

ESN-SVM

Esta metodología conserva el aprendizaje no-supervisado de la representación mediante una ESN. La diferencia con la metodología anterior radica en el uso de un modelo basado en SVM que reemplaza a VAE para el aprendizaje desde un solo estado. Este modelo es capaz de aprender una frontera de decisión en el espacio de características dado por los pesos de ESN, la cual delimita el contorno de la distribución de datos perteneciente al estado conocido (aprendido).

Con el propósito de encontrar el mejor modelo One-Class SVM y realizar una comparación justa con ESN-VAE se elije el kernel no-lineal *función de base radial* (RBF, por sus siglas en inglés). Además, se realiza un proceso de optimización por malla de búsqueda de los parámetros ν y γ de la SVM. Cabe indicar que el parámetro ν representa la probabilidad permitida de encontrar una instancia del estado aprendido fuera de la frontera de decisión. Visto de otra manera, es una medida de distancia entre la frontera mínima posible obtenida con los datos de entrenamiento y la frontera de decisión final. Por otra parte. el parámetro γ es el coeficiente de la RBF.

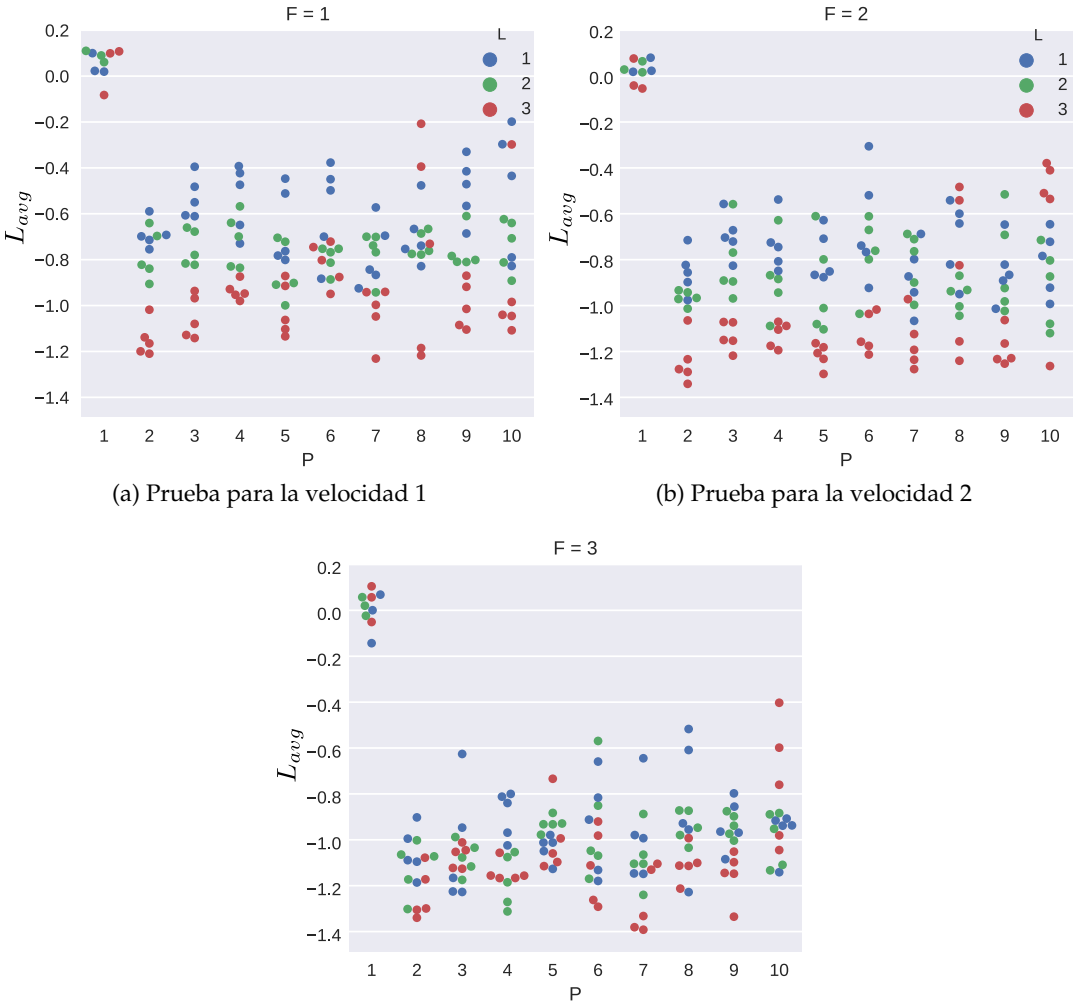
Los resultados obtenidos con ESN-SVM para diferentes valores de γ y ν se muestran en la Tabla 5.3, donde se presenta para cada combinación el error para la detección del estado normal[10]. Para el estado normal, este error se obtiene con la razón entre las instancias de estado normal que tienen un distancia menor a 0 y el número total de instancias de este estado. En el mismo sentido, el error en la detección de estados con falla se obtiene con la relación entre el número de instancias con falla con una distancia mayor a 0 y el número total de instancias con falla en el conjunto de prueba. Se puede apreciar que para todos los valores de γ, el menor error de detección del estado normal se obtiene con un ν pequeño. Esto indica que, al considerar los pesos de la capa de salida de ESN como variables deterministas, las instancias del estado normal se encuentran muy cercanas a las instancias del resto de estados en el espacio de características generado. Lo que es corroborado en la Figura 5.11, donde se muestra la distancia existente entre nuevas instancias y la frontera de decisión para diferentes estados.

Con el modelo obtenido, la separación entre el estado normal del resto se ve

[10]En todas las pruebas el otro error en todos los casos es siempre 0.0.

γ/ν	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	22.22	29.63	40.74	55.56	74.07	81.48	100.0	100.0	100.0
0.2	48.15	48.15	51.85	70.37	77.78	85.19	92.59	100.0	100.0
0.3	74.07	74.07	74.07	77.78	88.89	92.59	92.59	100.0	100.0

Cuadro 5.3: Errores de estado normal/estado con falla para valores de γ entre 0.1 y 0.3 y ν entre 0.1 y 0.9, ambos con incrementos de 0.1.



(a) Prueba para la velocidad 1

(b) Prueba para la velocidad 2

(c) Prueba para la velocidad 3

Figura 5.11: Distancia desde instancias con diferentes combinaciones de P-estados, F-velocidades y L-cargas hacia la frontera de decisión para el modelo ESN-SVM.

acentuada en mayor proporción a medida que la velocidad es incrementada. Este efecto también fue observado en el modelo ESN-VAE, de donde se podría deducir que para velocidades altas es posible cambiar el umbral de decisión de 0 a otro valor que permita la separación de los estados. Sin embargo, no parece muy recomendable realizar este procedimiento en función de los resultados obtenidos en el conjunto de prueba.

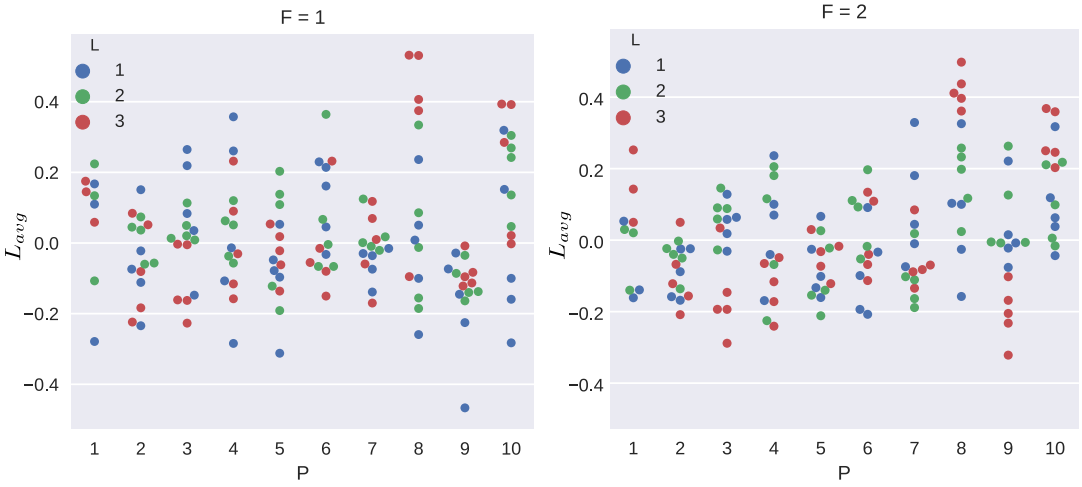
Adicionalmente, del análisis de la Tabla 5.3 se deduce que, para este caso concreto, un valor pequeño en los parámetros de SVM genera un modelo con mejor desempeño que el resto. Por esta razón es necesario realizar una optimización exhaustiva por debajo de 0.1 para los dos parámetros con decrementos de 0.01. Como resultado de este procedimiento se obtiene un error de detección del estado normal de 14.82, y 4.20 para la detección del resto de estados con una configuración de $\nu = 0{,}05$ y $\gamma = 0{,}05$, alcanzando con esto el mejor modelo posible de entre los del test. Sin embargo, los resultados que se obtienen con ESN-VAE dan un error de 0.0 para los dos casos de detección (estado normal y el resto de estados) dando a ESN-VAE una ventaja contundente frente a ESN-SVM en lo que a desempeño se refiere.

TS-SVM

En est experimento se descarta el uso de ESN para la transformación desde series temporales a un nuevo espacio de representación. Aquí en cambio se cataloga cada elemento de la serie como una característica del estado del sistema dinámico. Sin embargo, es conveniente hacer notar que, como en la mayoría de procesos, aquí no se dispone de una señal de sincronización[11] de la serie temporal y tampoco se ha identificado la duración de un periodo del proceso. Por lo que se toman de manera consecutiva series de tiempo de un tamaño específico, que son pasadas al modelo de One-Class Learning basado en SVM para la identificación de una frontera de decisión.

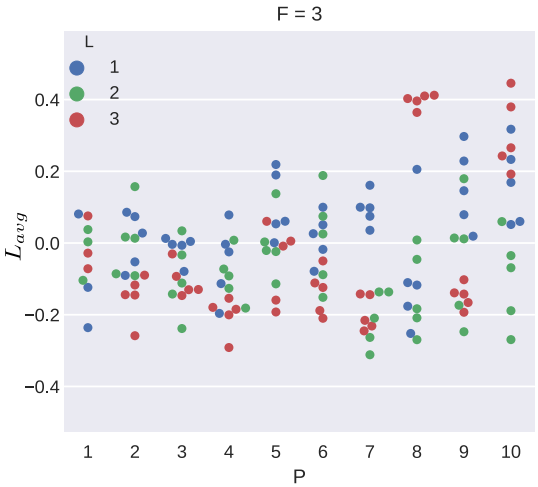
Los resultados obtenidos con un tamaño de 1000 para cada serie de tiempo son: error de detección de estado normal $= 37{,}04\,\%$, y error de detección de estados con falla $= 43{,}95\,\%$. Los resultados de la evaluación del modelo resultante para diferentes velocidades y cargas puede ser visto en la Figura 5.12. Como se puede apreciar, los resultados de esta metodología no son comparables con ESN-VAE o ESN-SVM, por lo que parece concluirse que la codificación realizada por ESN es determinante para el buen desempeño del modelo.

[11] Las señales de sincronización permiten identificar el periodo completo de un proceso, como puede ser una revolución de un motor.



(a) Prueba para la velocidad 1

(b) Prueba para la velocidad 2



(c) Prueba para la velocidad 3

Figura 5.12: Distancia desde instancias con diferentes combinaciones de P-estados, F-velocidades y L-cargas hacia la frontera de decisión para el modelo TS-SVM.

CAPÍTULO 6

CONCLUSIONES

Aunque muchas de las conclusiones que se pueden extraer de las diferentes aproximaciones que se muestran en esta memoria se han ido intercalando en las secciones de los capítulos anteriores, que conforman el núcleo de la investigación desarrollada, queremos cerrar nuestro trabajo con un breve repaso de los avances realizados con el fin principal de dar una visión global y transversal de las ideas, aproximaciones, resultados y comparativas obtenidas.

Como se ha puesto de manifiesto en las páginas precedentes, el espíritu fundamental de esta tesis era el de fundamentar, proponer, e implementar un conjunto de metodologías para la creación, entrenamiento, e inferencia de modelos de Aprendizaje Automático que tengan como objetivo caracterizar el comportamiento de Sistemas Dinámicos a partir de grupos de señales (posiblemente, de alta complejidad) medidas en ellos. Queremos resaltar que el concepto de señal se entiende de la forma más general posible: una sucesión uniforme de mediciones que evolucionan en el tiempo.

Los enfoques descritos en este trabajo aúnan técnicas provenientes del Procesamiento de Señales junto con aquellas más habituales en el área del Aprendizaje Automático. A lo largo de la investigación hemos comprobado cómo, partiendo de unas primeras aproximaciones que combinan técnicas de ambas áreas junto con el conocimiento explícito de los expertos, se han ido desarrollando metodologías más elaboradas y complejas hasta proponer modelos basados en datos que procesan las señales de forma completamente automática. En este sentido, la evolución mostrada en este trabajo ha surgido de forma metódica a partir de las deficiencias que se han ido encontrando en las diversas aproximaciones, y que ya eran patentes en otras aproximaciones anteriores (no necesariamente haciendo uso de Aprendizaje Automático).

Junto a esta evolución metodológica, también es apreciable una evolución paralela en los modelos de aprendizaje utilizados, que comienza con el modelo de Random Forest (del año 1999, y que hoy en día puede ser considerado tan popular como clásico), continúa con una versión extendida de las Redes Neuronales Convolucionales,

perteneciente a los inicios del Deep Learning (año 2011), y culmina con un modelo híbrido que hace uso de una red neuronal recurrente, la Echo State Network, junto con un modelo variacional, el Variational AutoEncoder, como dignos representantes del actual estado del arte de la Computación con Reservorios y el Deep Learning (2016). Este último modelo no solo proporciona una útil herramienta para independizar el trabajo del modelo del trabajo del experto, sino que además consigue una primera aproximación a un generador automático de señales aprendidas, un tópico del aprendizaje que hoy en día promete tantas aplicaciones en el mundo real como las aportadas por la clasificación y la predicción.

Queremos destacar también aquí la aplicación al mundo real que ha servido de referencia y validador de los modelos que se iban generando. Por una parte, el utilizar un problema de referencia general y transversal en toda la tesis (el del Mantenimiento Basado en la Condición) nos obliga a no teorizar en el vacío acerca de la bondad de los diversos modelos y aproximaciones generadas, algo desgraciadamente muy común en trabajos similares. Por otra, y a pesar de que el contexto de trabajo se ha limitado a ese problema, representa realmente un marco general de clasificación y predicción de sistemas dinámicos en los que podemos encontrar dos factores determinantes para la correcta valoración del trabajo desarrollado: opera sobre sistemas dinámicos que muestran comportamientos muy complejos y que se han resistido a un modelado clásico, y tienen el suficiente interés económico como para que se hayan desarrollado sistemas de adquisición de datos con una calidad que solo se pueden encontrar en otros pocos dominios. Además, el problema que se pretende resolver por medio de los modelos desarrollados es de una importancia fundamental no solo desde un punto de vista económico, sino también social (por el incremento de seguridad que supone poder anticipar posibles fallos futuros en cualquier maquinaria).

Concretamente, la primera metodología que se ha propuesto en esta memoria caracteriza los estados de un sistema dinámico con la energía de las subseñales obtenidas por la descomposición con Wavelet Packet de cada señal original capturada en los estados específicos que se quieren modelar. El conjunto de familias candidato para optimizar esta descomposición se ha elegido a partir del análisis de trabajos previos en el estado del arte de los sistemas dinámicos concretos que se han modelado. Estos datos alimentan a un modelo de Random Forest que es optimizado en sus hiper-parámetros por medio de un algoritmo voraz (de búsqueda semi-exhaustiva). Las mejores familias de wavelet son encontradas para el sistema dinámico específico así como también las mejores características (energías) dentro de las familias seleccionadas. Con esta información filtrada, finalmente se construye un clasificador, igualmente basado en RF, usando los parámetros optimizados.

Esta metodología muestra ventajas evidentes, como es la simplicidad a la hora de construcción del modelo, unos excelentes resultados en diferentes tareas del mantenimiento basado en la condición, y una alta flexibilidad en la aplicación de la metodología en diferentes sistemas dinámicos. Sin embargo, son observables algunas deficiencias que no podemos pasar por alto:

1. La metodología solo es aplicable si se conocen los estados a los que pertenecen las señales de entrenamiento (enfoque supervisado puro). En consecuencia, su aplicabilidad en tipos de problemas se limita considerablemente.

2. Solo se puede aplicar si se tienen señales de todos los estados que se desea modelar. Es decir, que requiere disponer de un muestrario suficientemente alto de comportamientos posibles del sistema, incluso aquellos no deseables, incontrolables, o excesivamente caros (en el sentido de que conllevan la degradación del sistema).

3. Es necesario un conocimiento a priori de las familias de wavelet que mejor descomponen las señales del sistema dinámico con el que se está lidiando. En consecuencia, depende de un conocimiento experto previo del sistema, algo que puede estar fuera de nuestro alcance en muchas situaciones, bien porque resulte excesivamente caro, o difícil, disponer de un experto en el tema; bien porque el sistema plantee interrogantes desconocidos que ningún experto puede resolver.

La segunda metodología desarrollada en esta memoria aborda la tercera deficiencia señalada anteriormente. Es decir, tiene como objetivo eliminar la necesidad de un conocimiento a priori de los mejores kernel para descomponer un tipo específico de señal, y plantea un enfoque generalizado para la caracterización no-supervisada de los estados de un sistema dinámico desde una versión en el dominio de tiempo-frecuencia de las señales. Para ello, la metodología en cuestión funciona en un proceso conformado por dos etapas de aprendizaje. En la primera de ellas se utiliza la idea del autoencoder adaptada para capas convolucionales, lo que permite obtener un conjunto óptimo de kernels (con función similar a lo que haría una familia de wavelets) capaz de codificar los datos de entrada en un espacio de representación reducido, desde el cual se puede reconstruir nuevamente la entrada. En la segunda etapa optimiza ese conjunto de kernels en una red convolucional profunda con la información brindada por los estados específicos de las señales de entrenamiento. La conjunción de estas dos etapas logra automatizar el proceso de extracción de características de una serie de tiempo.

La aplicación de esta propuesta al análisis de sistemas dinámicos se pone de manifiesto en la aplicación mostrada a la evaluación de la severidad de daño, ya que muestra claras ventajas en cuanto a la exactitud de clasificación de estados frente a otras propuestas disponibles en la literatura usadas para el mismo fin de extracción no-supervisada de características desde series temporales. Es decir, el modelo generado no solo funciona bien en la clasificación discreta de estados, sino como predictor de severidad del daño (estado interno, y no medible, del sistema). Adicionalmente, presenta una ventaja adicional que proporciona indicios de una correcta extrapolación de la metodología desarrollada a otros análisis similares de sistemas dinámicos generales, y es que en este caso no se asumió ningún tipo específico de kernels, sino que éstos son aprendidos a partir de los datos con el fin de optimizar el desempeño de la tarea encargada.

A pesar de la indudable mejora que esta nueva metodología supone respecto de la anterior (y las más clásicas y habituales en el análisis de sistemas dinámicos desde un punto de vista de los datos), se mantienen las dos primeras deficiencias observadas en la primera metodología: el sistema de aprendizaje sigue siendo supervisado, y requiere de la adquisición de señales en una muestra grande de comportamientos posibles del sistema. Lo que continúa siendo muy costoso desde la perspectiva de adquisición de datos.

Con el fin de mitigar las deficiencias observadas se desarrolla una tercera aproximación que, sin lugar a dudas, supone la aportación más novedosa de este trabajo. A pesar de esta valoración, hemos de indicar que su aplicabilidad no se produce en los mismos contextos que las anteriores, por lo que no subsume los resultados mostrados en los capítulos anteriores.

Como tercera propuesta se ha desarrollado un método novedoso para el modelado de señales provenientes de un estado específico de un sistema dinámico. Desde el punto de vista del Aprendizaje Automático, este problema se corresponde con lo que se denomina "One-Class Learning Problem", pues se supone la posibilidad de conseguir datos de una sola clase (en el funcionamiento del sistema, representaría el estado de funcionamiento normal, del que resulta factible y barato adquirir señales), y se desea disponer de un modelo que caracterice esta clase sin necesidad de haber visto los otros modos de operación del sistema. Como hemos observado en el capítulo correspondiente, esta aproximación es esencialmente distinta, ya que no estamos en este caso ante un problema de clasificación supervisada, puesto que no disponemos de ejemplares de varias clases entre los que poder aprender diferencias caracterizadoras, tampoco estamos ante un problema de clusterización que pretende aprender a agrupar en bolsas distintas una colección de muestras que se suponen distintas. De lo único que disponemos es de una colección de muestras que sabemos que reflejan una característica común del sistema y, tras el entrenamiento adecuado, queremos que el modelo aprendido sea capaz de reconocer cuándo una nueva muestra (no usada para el aprendizaje) tiene características que la hacen esencialmente distinta de las vistas anteriormente.

En este sentido, las muestras que podemos utilizar deben distribuirse siguiendo algún patrón reconocible en el espacio de señales, por lo que la tarea de reconocer dicho patrón se relaciona estrechamente con el cálculo de la distribución de probabilidad compleja que, supuestamente, siguen nuestras muestras. Debido a que las señales en bruto son objetos de un espacio difícilmente manipulables y poco apropiados para un proceso de aprendizaje automático, el primer paso consiste en obtener una representación de las mismas de forma no supervisada, para lo que hemos hecho uso de una Echo State Network como codificadora, un tipo muy particular de red recurrente especialmente apropiada para el tratamiento de señales provenientes de una dinámica. A pesar de este cambio de representación, el problema original de obtener la distribución de las muestras sigue siendo el mismo, y hemos hecho uso de Variational Autoencoder, una versión estocástica del autocodificador neuronal clásico capaz de modelar este tipo de distribuciones a partir de datos. Además, se propone el valor del error de reconstrucción como una métrica informativa para

discriminar entre el estado normal modelado y el resto de estados posibles del sistema, donde un pequeño error representa pertenencia a la clase aprendida y un error grande representa algún otro estado que no sigue la distribución inferida.

Este método propuesto presenta dos ventajas principales en relación a los métodos anteriores:

1. Solo necesita señales del estado que se desea modelar, que son fáciles de obtener en la mayoría de sistemas dinámicos.

2. A diferencia de otros métodos basados en señales donde un conocimiento específico previo del sistema mecánico a analizarse es necesario, el método propuesto no requiere del conocimiento experto debido a la etapa de aprendizaje no supervisado de la representación.

Este método ha sido aplicado a la tarea de detección de fallas en maquinaria rotativa donde el modelo se construye únicamente con señales de vibración sin fallo en diferentes condiciones de funcionamiento. El objetivo es, una vez entrenado un modelo para un sistema en funcionamiento normal, que el modelo reconozca cuándo se produce un cambio en el funcionamiento del sistema debido a la aparición de modificaciones internas (fallos) de alguno/s de los componentes del sistema. Las comparaciones mostradas en el capítulo con otro método reportado en la literatura (hay pocos modelos de este tipo con el que poder realizar comparaciones) muestran una clara ventaja de nuestra propuesta, tanto en exactitud como automatización en la fase de caracterización de estados.

Con esta última aproximación se han conseguido muchos de los objetivos marcados inicialmente en la investigación que ha dado lugar a esta tesis, ya que se han conseguido superar las deficiencias 2 y 3 puestas de manifiesto en la primera metodología (y comunes a la mayoría de las metodologías actuales de análisis de sistemas dinámicos) y de forma parcial (casi completa) la deficiencia 1, debido a que todavía es necesario garantizar que las señales con las que se construye el modelo pertenecen al estado que se desea caracterizar en el sistema dinámico.

Por supuesto, casi ningún trabajo de investigación puede considerarse definitivo ni completo, y a partir de las consideraciones y aproximaciones realizadas se han abierto nuevas vías de trabajo para futuras investigaciones, tanto teóricas como aplicadas, que vienen a abordar direcciones que no hemos considerado o a completar líneas consideradas que merecen ser estudiadas con mayor atención.

Por supuesto, y no obviamos de ninguna de las formas una gran tarea pendiente en nuestro trabajo, está la de poner en práctica los modelos desarrollados sobre sistemas dinámicos de dominios distintos a los considerados aquí. A pesar de que, como hemos resaltado anteriormente, los sistemas dinámicos considerados en el problema del mantenimiento basado en la condición son muy generales y ricos, no pasamos por alto que las aproximaciones presentadas en esta memoria pueden estar sesgadas por el tipo de problema abordado, por lo que sería de gran interés comprobar su eficacia en otros sistemas dinámicos, como son: señales provenientes de sistemas de

salud, reconocimiento de patrones en señales medidas en nodos de un red dinámica (por ejemplo, redes eléctricas, de comunicaciones, etc.), sistemas meteorológicos, etc. En consecuencia, una de las primeras tareas a llevar a cabo tras esta etapa será la validación, y posible adecuación, de las aproximaciones mostradas a escenarios más generales.

Desde un punto de vista más teórico, queda pendiente el modelado totalmente no-supervisado de sistemas dinámicos con aprendizaje automático sin ningún conocimiento sobre los estados a los que pertenecen las señales capturadas, lo que supondría una continuación lógica para abordar completamente la primera deficiencia de nuestra primera aproximación. Para ello, sería necesaria una interpretación de la zonas ocupadas por las señales provenientes de distintos estados en el espacio de probabilidad generado en la fase de aprendizaje de la representación en la tercera propuesta.

Otra línea de trabajo futuro que resulta de un interés práctico indudable sería la construcción de modelos capaces de aprender de forma incremental nuevos estados del sistema dinámico sin degradar lo aprendido previamente del estado inicial y con la capacidad de interpolar este conocimiento hacia otros estados intermedios (por ejemplo degradación intermedia entre dos estados conocidos de un componente mecánico). Dentro del mantenimiento basado en la condición un trabajo que podría ser abordado en un futuro es la detección temprana y el pronóstico de cambio de estados en un sistema mecánico a partir de mediciones del proceso, lo que permitiría una planificación del mantenimiento basado en estados reales actuales y futuros desembocando en un manejo eficiente de los recursos adquisitivos de una empresa.

En general, la implantación de sistemas, como el aquí desarrollado, en entornos de trabajo reales sería de un gran valor para la evolución de las metodologías, además de (esperamos) un valor añadido al funcionamiento de muchas implementaciones del ámbito de la ingeniería. A pesar de la calidad de los bancos de pruebas y de los sistemas de adquisición de datos, es innegable que el mundo real proporciona un entorno de trabajo mucho más exigente y complejo.

Apéndices

APÉNDICE A

PUBLICACIONES

Los resultados del proceso de investigación realizado en esta tesis se han difundido a la comunidad científica utilizando los medios tradicionales, mediante publicaciones en revistas indexadas tanto en las bases de datos Scopus como también en ISI-Web of Science.

A continuación se presenta la lista de trabajos publicados en las áreas de Aprendizaje Automático y/o Procesamiento de Señales aplicados al modelado de sistemas dinámicos y mantenimiento basado en la condición que están relacionados de forma directa o indirecta con esta tesis:

- Diego Cabrera, Fernando Sancho, René-Vinicio Sánchez, Grover Zurita, Mariela Cerrada, Chuan Li, and RafaelE. Vásquez. Fault diagnosis of spur gearbox based on random forest and wavelet packet decomposition. *Frontiers of Mechanical Engineering*, pages 1–10, 2015
- Diego Cabrera, Fernando Sancho, Chuan Li, Mariela Cerrada, René-Vinicio Sánchez, Fannia Pacheco, and José Valente de Oliveira. Automatic feature extraction of time-series applied to fault severity assessment of helical gearbox in stationary and non-stationary speed operation. *Applied Soft Computing*, 58:53–64, sep 2017
- Diego Cabrera, Fernando Sancho, and Felipe Tobar. Combining reservoir computing and variational inference for efficient one-class learning on dynamical systems. In *Sensing, Diagnostics, Prognostics and Control (SDPC), 2017 International Conference on*. IEEE, 2017
- Chuan Li, René-Vinicio Sanchez, Grover Zurita, Mariela Cerrada, Diego Cabrera, and Rafael E. Vásquez. Multimodal deep support vector classification with homologous features and its application to gearbox fault diagnosis. *Neurocomputing*, 168:119–127, nov 2015
- Mariela Cerrada, Grover Zurita, Diego Cabrera, René-Vinicio Sánchez, Mariano Artés, and Chuan Li. Fault diagnosis in spur gears based on genetic

algorithm and random forest. *Mechanical Systems and Signal Processing*, 70-71:87–103, mar 2016

- Mariela Cerrada, René Sánchez, Diego Cabrera, Grover Zurita, and Chuan Li. Multi-stage feature selection by using genetic algorithms for fault diagnosis in gearboxes based on vibration signal. *Sensors*, 15(9):23903–23926, sep 2015
- Chuan Li, René-Vinicio Sanchez, Grover Zurita, Mariela Cerrada, Diego Cabrera, and Rafael E. Vásquez. Gearbox fault diagnosis based on deep random forest fusion of acoustic and vibratory signals. *Mechanical Systems and Signal Processing*, 76-77:283–293, aug 2016
- Chuan Li, Diego Cabrera, José Valente de Oliveira, René-Vinicio Sanchez, Mariela Cerrada, and Grover Zurita. Extracting repetitive transients for rotating machinery diagnosis using multiscale clustered grey infogram. *Mechanical Systems and Signal Processing*, 76-77:157–173, aug 2016
- Chuan Li, René-Vinicio Sánchez, Grover Zurita, Mariela Cerrada, and Diego Cabrera. Fault diagnosis for rotating machinery using vibration measurement deep statistical feature learning. *Sensors*, 16(6):895, jun 2016
- Chuan Li, Vinicio Sanchez, Grover Zurita, Mariela Cerrada Lozada, and Diego Cabrera. Rolling element bearing defect detection using the generalized synchrosqueezing transform guided by time–frequency ridge enhancement. *ISA Transactions*, 60:274–284, jan 2016
- Mariela Cerrada, René-Vinicio Sánchez, Fannia Pacheco, Diego Cabrera, Grover Zurita, and Chuan Li. Hierarchical feature selection based on relative dependency for gear fault diagnosis. *Applied Intelligence*, 44(3):687–703, nov 2015
- Fannia Pacheco, José Valente de Oliveira, René-Vinicio Sánchez, Mariela Cerrada, Diego Cabrera, Chuan Li, Grover Zurita, and Mariano Artés. A statistical comparison of neuroclassifiers and feature selection methods for gearbox fault diagnosis under realistic conditions. *Neurocomputing*, Feb 2016
- Chuan Li, José Valente de Oliveira, Mariela Cerrada, Fannia Pacheco, Diego Cabrera, Vinicio Sanchez, and Grover Zurita. Observer-biased bearing condition monitoring: From fault detection to multi-fault classification. *Engineering Applications of Artificial Intelligence*, 50:287–301, apr 2016
- Fannia Pacheco, Mariela Cerrada, René-Vinicio Sánchez, Diego Cabrera, Chuan Li, and José Valente de Oliveira. Attribute clustering using rough set theory for feature selection in fault severity classification of rotating machinery. *Expert Systems with Applications*, 71:69–86, apr 2017
- Chuan Li, José Valente de Oliveira, René-Vinicio Sanchez, Mariela Cerrada, Grover Zurita, and Diego Cabrera. Fuzzy determination of informative frequency band for bearing fault detection. *Journal of Intelligent & Fuzzy Systems*, 30(6):3513–3525, Apr 2016

- Mariela Cerrada, Chuan Li, René-Vinicio Sánchez, Fannia Pacheco, Diego Cabrera, and José Valente de Oliveira. A fuzzy transition based approach for fault severity prediction in helical gearboxes. *Fuzzy Sets and Systems*, dec 2016
- Fannia Pacheco, Mariela Cerrada, Rene Vinicio Sanchez, Diego Cabrera, Chuan Li, and Jose Valente de Oliveira. Clustering algorithm using rough set theory for unsupervised feature selection. In *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2016
- Mariela Cerrada, René-Vinicio Sánchez, Chuan Li, Fannia Pacheco, Diego Cabrera, José Valente de Oliveira, and Rafael E. Vásquez. A review on data-driven fault severity assessment in rolling bearings. *Mechanical Systems and Signal Processing*, 99:169–196, jan 2018
- Chuan Li, Luiz Ledo, Myriam Delgado, Mariela Cerrada, Fannia Pacheco, Diego Cabrera, René-Vinicio Sánchez, and José Valente de Oliveira. A bayesian approach to consequent parameter estimation in probabilistic fuzzy systems and its application to bearing fault classification. *Knowledge-Based Systems*, 129:39–60, aug 2017
- Fannia Pacheco, Mariela Cerrada, Rene Vinicio Sanchez, Diego Cabrera, Chuan Li, and Jose Valente de Oliveira. A methodological framework using statistical tests for comparing machine learning based models applied to fault diagnosis in rotating machinery. In *2016 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE, nov 2016

APÉNDICE B

SOFTWARE

Para la prueba de concepto de las metodologías propuestas en este trabajo, se ha desarrollado software específico para cada una de ellas. El lenguaje de programación común para estas implementaciones es Python (Python Software Fundation, `https://www.python.org/`) en su versión 3.5. Sin embargo para cada una de las metodologías se trabajó con librerías distintas dependiendo del soporte y funcionalidades presentes al momento de la implementación.

A continuación se dan detalles de cada una ellas.

B.1. Extracción de Características y Clasificación con un Enfoque Clásico

La implementación de esta metodología se encuentra en el repositorio `https://github.com/diegoroman17/RF_methodology` y consta de tres archivos de código fuente principales para su ejecución:

- `src/feature_extraction_wpd.py`
- `src/optimizer_parameters.py`
- `test/methodology.ipynb`

El script `feature_extraction_wpd.py` es el encargado de procesar las señales capturadas del sistema dinámico que han sido almacenadas en la base de datos con el fin de realizar el proceso de extracción de características mediante la descomposición por wavelet packet. Esta descomposición es llevada a cabo mediante la librería **PyWavelets** [2]. Como resultado, un objeto de tipo `Data_set` con información acerca del proceso de extracción de características es almacenado mediante serialización en un archivo con extensión `.pickle`, lo cual se logra con la librería **Pickle** de Python 3 (ver `https://docs.python.org/3/library/pickle.html` para más detalles).

El script `optimizer_parameters.py` lee el objeto anteriormente almacenado y crea clasificadores de tipo Random Forest con cada posible parámetro y conjunto de familias de wavelet. Estos clasificadores a su vez son evaluados usando oob-error como métrica. La creación de estos modelos es llevada a cabo mediante la librería **Scikit-learn** [80], muy popular en el área. Los resultados anteriores son puestos en un objeto, el cual es almacenado igualmente por serialización en un archivo `.pickle`.

El notebook `methodology.ipynb` contiene el resto de la metodología, la cual es ejecutada a partir de la lectura del archivo almacenando anteriormente. Este notebook además muestra los resultados para cada conjunto de parámetros de forma gráfica para un mejor análisis.

B.2. Aprendizaje de Características de Series de Tiempo

La implementación de esta metodología se encuentra en el repositorio `https://github.com/diegoroman17/SCAE_methodology` y está compuesto de dos archivos principales:

- `src/preprocessing.py`
- `src/SCAE.py`

El script `preprocessing.py` se encarga de transformar al dominio tiempo-frecuencia las series de tiempo almacenadas en la base de datos creada por el sistema de adquisición. Esta transformación es llevada a cabo usando la librería **PyWavelets** como en el caso anterior. El resultado es un conjunto de tuplas tipo $(imagen, etiqueta)$ las cuales son agrupadas en un arreglo de **Numpy** [95], una estructura muy eficiente para cómputo numérico. Esta estructura es almacenada para la próxima etapa.

El script `SCAE.py` contiene el núcleo de esta metodología. Aquí se crea un grafo de computación con el modelo Stacked Convolutional AutoEncoder para luego ser optimizado usando el algoritmo de gradiente descendente estocástico. Al ser un modelo de Deep Learning, lo anterior se logra con **Theano** [4], una de las primeras librerías en el área de Deep Learning y además entre las más populares.

B.3. Aprendizaje de la Representación y One-Class Learning

La implementación de esta metodología se encuentra en el repositorio `https://github.com/diegoroman17/ESN_VAE_methodology` y está compuesto de dos archivos principales:

- `src/esn_features.py`
- `src/vae_modified.py`

El script `esn_features.py` realiza el aprendizaje de la representación con la Echo State Network de cada serie de tiempo del estado que se desea modelar, almacenada en la base de datos que fue creada con anterioridad por el sistema de adquisición. Debido a que este proceso consume una gran cantidad de tiempo, se utilizó la librería para procesamiento distribuido de Python llamada **Multiprocessing** (ver `https://docs.python.org/3/library/multiprocessing.html` para más información), lo que permite que se utilicen todos los núcleos del computador reduciendo el tiempo de procesamiento en este mismo factor. El resultado es un arreglo de datos con cada elemento almacenando la nueva representación de una serie de tiempo.

Posteriormente con el script `vae_modified.py` se realiza el entrenamiento del Variational AutoEncoder para el modelado de la distribución de probabilidad compleja de los datos anteriormente almacenados. Aquí también se realiza la evaluación del modelo con los datos de prueba y se imprimen las métricas de clasificación. La implementación de este modelo se ha realizado utilizando la librería **Tensorflow** [3] creada y liberada por Google Inc. para el desarrollo de modelos de Deep Learning y computación de alto rendimiento en general.

Bibliografía

[1] Mezcla de lubricantes: Garantía de problemas. http://noria.mx/lublearn/mezcla-de-lubricantes-garantia-de-problemas/. Accessed: 2017-01-06.

[2] Pywavelets - wavelet transforms in python. https://pywavelets.readthedocs.io/en/latest/. Accessed: 2017-09-18.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre-Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron Courville, Yann N. Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman, Laurent Dinh, Mélanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian Goodfellow, Matt Graham, Caglar Gulcehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrancois, Si-

mon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert T. McGibbon, Roland Memisevic, Bart van Merriënboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Pal, Razvan Pascanu, Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schlüter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, and Ying Zhang. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[5] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, Jun 1993.

[6] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, oct 1997.

[7] William A. Belson. Matching and prediction on the principle of biological classification. *Applied Statistics*, 8(2):65, jun 1959.

[8] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory - COLT 92*. ACM Press, 1992.

[9] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[10] Leo Breiman. Random forests. *UC Berkeley TR567*, pages 123–140, 1999.

[11] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[12] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *IEEE TRANSACTIONS ON COMMUNICATIONS*, 31:532–540, 1983.

[13] Diego Cabrera, Fernando Sancho, Chuan Li, Mariela Cerrada, René-Vinicio Sánchez, Fannia Pacheco, and José Valente de Oliveira. Automatic feature extraction of time-series applied to fault severity assessment of helical gearbox in stationary and non-stationary speed operation. *Applied Soft Computing*, 58:53–64, sep 2017.

[14] Diego Cabrera, Fernando Sancho, René-Vinicio Sánchez, Grover Zurita, Mariela Cerrada, Chuan Li, and RafaelE. Vásquez. Fault diagnosis of spur gearbox based on random forest and wavelet packet decomposition. *Frontiers of Mechanical Engineering*, pages 1–10, 2015.

[15] Diego Cabrera, Fernando Sancho, and Felipe Tobar. Combining reservoir computing and variational inference for efficient one-class learning on dynamical systems. In *Sensing, Diagnostics, Prognostics and Control (SDPC), 2017 International Conference on*. IEEE, 2017.

[16] Mariela Cerrada, Chuan Li, René-Vinicio Sánchez, Fannia Pacheco, Diego Cabrera, and José Valente de Oliveira. A fuzzy transition based approach for fault severity prediction in helical gearboxes. *Fuzzy Sets and Systems*, dec 2016.

[17] Mariela Cerrada, René Sánchez, Diego Cabrera, Grover Zurita, and Chuan Li. Multi-stage feature selection by using genetic algorithms for fault diagnosis in gearboxes based on vibration signal. *Sensors*, 15(9):23903–23926, sep 2015.

[18] Mariela Cerrada, René-Vinicio Sánchez, Chuan Li, Fannia Pacheco, Diego Cabrera, José Valente de Oliveira, and Rafael E. Vásquez. A review on data-driven fault severity assessment in rolling bearings. *Mechanical Systems and Signal Processing*, 99:169–196, jan 2018.

[19] Mariela Cerrada, René-Vinicio Sánchez, Fannia Pacheco, Diego Cabrera, Grover Zurita, and Chuan Li. Hierarchical feature selection based on relative dependency for gear fault diagnosis. *Applied Intelligence*, 44(3):687–703, nov 2015.

[20] Mariela Cerrada, Grover Zurita, Diego Cabrera, René-Vinicio Sánchez, Mariano Artés, and Chuan Li. Fault diagnosis in spur gears based on genetic algorithm and random forest. *Mechanical Systems and Signal Processing*, 70-71:87–103, mar 2016.

[21] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, jan 1967.

[22] A Criminisi and J Shotton. Introduction: The abstract forest model. In *Decision Forests for Computer Vision and Medical Image Analysis*, pages 7–23. Springer, 2013.

[23] Stephen P Curram and John Mingers. Neural networks, decision tree induction and discriminant analysis: An empirical comparison. *Journal of the Operational Research Society*, 45(4):440–450, 1994.

[24] Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909–996, Oct 1988.

[25] Carl Doersch. Tutorial on variational autoencoders. June 2016.

[26] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[27] Zhipeng Feng and Ming J. Zuo. Fault diagnosis of planetary gearboxes via torsional vibration signal analysis. *Mechanical Systems and Signal Processing*, 36(2):401–421, Apr 2013.

[28] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[29] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[30] Dennis Gabor. Theory of communication. *J. Inst. Elect. Eng.*, 93:429–457, 1946.

[31] M. Y. Gokhale and Daljeet Kaur Khanduja. Time domain signal analysis using wavelet packet decomposition approach. *International Journal of Communications, Network and System Sciences*, 03(03):321–329, 2010.

[32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[33] K.C. Gryllias and I.A. Antoniadis. A support vector machine approach based on physical model training for rolling element bearing fault detection in industrial environments. *Engineering Applications of Artificial Intelligence*, 25(2):326–344, mar 2012.

[34] W. Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. *Zeitschrift für Physik*, 43(3-4):172–198, Mar 1927.

[35] G. E. Hinton. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, Jul 2006.

[36] Geoffrey E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1-3):185–234, Sep 1989.

[37] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, nov 1997.

[38] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, Jan 1991.

[39] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, jan 1989.

[40] Sajid Hussain and Hossam A. Gabbar. Gearbox fault detection using real coded genetic algorithm and novel shock response spectrum features extraction. *Journal of Nondestructive Evaluation*, nov 2013.

[41] H. Jaeger. Adaptive nonlinear system identification with echo state networks. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 593–600. MIT Press, 2002.

[42] Andrew K.S. Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483–1510, oct 2006.

[43] I.T. Jolliffe. *Principal Component Analysis (Springer Series in Statistics)*. Springer, 2002.

[44] Seifedine Kadry, editor. *Diagnostics and Prognostics of Engineering Systems*. IGI Global, 2013.

[45] Derek Kanneg and Wilson Wang. A wavelet spectrum technique for machinery fault diagnosis. *Journal of Signal and Information Processing*, 02(04):322–329, 2011.

[46] Nour El Islem Karabadji, Ilyes Khelf, Hassina Seridi, and Lakhdar Laouar. Genetic optimization of decision tree choice for fault diagnosis in an industrial ventilator. In *Condition Monitoring of Machinery in Non-Stationary Operations*, pages 277–283. Springer Berlin Heidelberg, 2012.

[47] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[48] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[49] Chuan Li, Diego Cabrera, José Valente de Oliveira, René-Vinicio Sanchez, Mariela Cerrada, and Grover Zurita. Extracting repetitive transients for rotating machinery diagnosis using multiscale clustered grey infogram. *Mechanical Systems and Signal Processing*, 76-77:157–173, aug 2016.

[50] Chuan Li, José Valente de Oliveira, Mariela Cerrada, Fannia Pacheco, Diego Cabrera, Vinicio Sanchez, and Grover Zurita. Observer-biased bearing condition monitoring: From fault detection to multi-fault classification. *Engineering Applications of Artificial Intelligence*, 50:287–301, apr 2016.

[51] Chuan Li, José Valente de Oliveira, René-Vinicio Sanchez, Mariela Cerrada, Grover Zurita, and Diego Cabrera. Fuzzy determination of informative frequency band for bearing fault detection. *Journal of Intelligent & Fuzzy Systems*, 30(6):3513–3525, Apr 2016.

[52] Chuan Li, Luiz Ledo, Myriam Delgado, Mariela Cerrada, Fannia Pacheco, Diego Cabrera, René-Vinicio Sánchez, and José Valente de Oliveira. A bayesian approach to consequent parameter estimation in probabilistic fuzzy systems and its application to bearing fault classification. *Knowledge-Based Systems*, 129:39–60, aug 2017.

[53] Chuan Li and Ming Liang. Time-frequency signal analysis for gearbox fault diagnosis using a generalized synchrosqueezing transform. *Mechanical Systems and Signal Processing*, 26:205–217, Jan 2012.

[54] Chuan Li, René-Vinicio Sánchez, Grover Zurita, Mariela Cerrada, and Diego Cabrera. Fault diagnosis for rotating machinery using vibration measurement deep statistical feature learning. *Sensors*, 16(6):895, jun 2016.

[55] Chuan Li, René-Vinicio Sanchez, Grover Zurita, Mariela Cerrada, Diego Cabrera, and Rafael E. Vásquez. Multimodal deep support vector classification with homologous features and its application to gearbox fault diagnosis. *Neurocomputing*, 168:119–127, nov 2015.

[56] Chuan Li, René-Vinicio Sanchez, Grover Zurita, Mariela Cerrada, Diego Cabrera, and Rafael E. Vásquez. Gearbox fault diagnosis based on deep random forest fusion of acoustic and vibratory signals. *Mechanical Systems and Signal Processing*, 76-77:283–293, aug 2016.

[57] Chuan Li, Vinicio Sanchez, Grover Zurita, Mariela Cerrada Lozada, and Diego Cabrera. Rolling element bearing defect detection using the generalized synchrosqueezing transform guided by time–frequency ridge enhancement. *ISA Transactions*, 60:274–284, jan 2016.

[58] Hui Li, Yuping Zhang, and Haiqi Zheng. Gear fault detection and diagnosis under speed-up condition based on order cepstrum and radial basis function neural network. *Journal of Mechanical Science and Technology*, 23(10):2780–2789, oct 2009.

[59] Zhixiong Li, Xinping Yan, Zhe Tian, Chengqing Yuan, Zhongxiao Peng, and Li Li. Blind vibration component separation and nonlinear feature extraction applied to the nonstationary vibration signals for the gearbox multi-fault diagnosis. *Measurement*, 46(1):259–271, Jan 2013.

[60] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, jan 2011.

[61] L. I. Rozonoer M. A. Aizerman, E. M. Braverman. Method of potential functions in the problem on restoration of functional converter characteristic by means of points observed randomly. *Avtomat. i Telemekh.*, 25(12):1705–1714, 1964.

[62] W. Maass. *Motivation, theory, and applications of liquid state machines*. Imperial College Press, 2011.

[63] S.G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, Jul 1989.

[64] Stephane Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. Academic Press, 2008.

[65] Manolis Maragoudakis, Euripides Loukis, and Panagiotis-Prodromos Pantelides. Random forests identification of gas turbine faults. In *Systems Engineering, 2008. ICSENG'08. 19th International Conference on*, pages 127–132. IEEE, 2008.

[66] G.N. Marichal, Mariano Artés, J.C. García Prada, and O. Casanova. Extraction of rules for faulty bearing classification by a neuro-fuzzy approach. *Mechanical Systems and Signal Processing*, 25(6):2073–2082, aug 2011.

[67] Donald W. Marquardt and Ronald D. Snee. Ridge regression in practice. *The American Statistician*, 29(1):3, feb 1975.

[68] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I*, ICANN'11, pages 52–59, Berlin, Heidelberg, 2011. Springer-Verlag.

[69] MATLAB. *version 9.2.0 (R2017a)*. The MathWorks Inc., Natick, Massachusetts, 2017.

[70] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, Dec 1943.

[71] L. Mdlazi, Tshilidzi Marwala, C. J. Stander, C. Scheffer, and P. S. Heyns. Principal component analysis and automatic relevance determination in damage identification. *CoRR*, abs/0705.1672, 2007.

[72] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[73] Nils Nilsson. *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*. McGraw-Hill, 1965.

[74] Fannia Pacheco, Mariela Cerrada, Rene Vinicio Sanchez, Diego Cabrera, Chuan Li, and Jose Valente de Oliveira. Clustering algorithm using rough set theory for unsupervised feature selection. In *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2016.

[75] Fannia Pacheco, Mariela Cerrada, Rene Vinicio Sanchez, Diego Cabrera, Chuan Li, and Jose Valente de Oliveira. A methodological framework using statistical tests for comparing machine learning based models applied to fault diagnosis in rotating machinery. In *2016 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE, nov 2016.

[76] Fannia Pacheco, Mariela Cerrada, René-Vinicio Sánchez, Diego Cabrera, Chuan Li, and José Valente de Oliveira. Attribute clustering using rough set theory for feature selection in fault severity classification of rotating machinery. *Expert Systems with Applications*, 71:69–86, apr 2017.

[77] Fannia Pacheco, José Valente de Oliveira, René-Vinicio Sánchez, Mariela Cerrada, Diego Cabrera, Chuan Li, Grover Zurita, and Mariano Artés. A statistical comparison of neuroclassifiers and feature selection methods for gearbox fault diagnosis under realistic conditions. *Neurocomputing*, Feb 2016.

[78] Mahesh Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005.

[79] Raj Kumar Patel and VK Giri. Feature selection and classification of mechanical fault of an induction motor using random forest classifier. *Perspectives in Science*, 8:334–337, 2016.

[80] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[81] Marcello Pelillo. Alhazen and the nearest neighbor rule. *Pattern Recognition Letters*, 38:34–37, mar 2014.

[82] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[83] Laura Elena Raileanu and Kilian Stoffel. Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1):77–93, 2004.

[84] Robert Bond Randall. *Vibration-based Condition Monitoring*. John Wiley & Sons, Ltd, jan 2011.

[85] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, 1962.

[86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, oct 1986.

[87] B. Samanta. Gear fault detection using artificial neural networks and support vector machines with genetic algorithms. *Mechanical Systems and Signal Processing*, 18(3):625–644, may 2004.

[88] Ulf D. Schiller and Jochen J. Steil. Analyzing the weight dynamics of recurrent learning algorithms. *Neurocomputing*, 63:5–23, jan 2005.

[89] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, jul 2001.

[90] George S. Sebestyen. *Decision-making processes in pattern recognition*. Macmillan New York, 1962.

[91] J. Stark, D.S. Broomhead, M.E. Davies, and J. Huke. Takens embedding theorems for forced and stochastic systems. *Nonlinear Analysis: Theory, Methods & Applications*, 30(8):5303–5314, dec 1997.

[92] John P. F. Sum, C.S. Leung, and L. W. Chan. Extended kalman filter in recurrent neural network training and pruning. Technical report, 1996.

[93] Floris Takens. *Detecting strange attractors in turbulence*, pages 366–381. Springer Berlin Heidelberg, Berlin, Heidelberg, 1981.

[94] V. Vakharia, V. K. Gupta, and P. K. Kankar. A comparison of feature ranking techniques for fault diagnosis of ball bearing. *Soft Computing*, 20(4):1601–1619, Feb 2015.

[95] Stefan van der Walt, S Chris Colbert, and Gael Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, mar 2011.

[96] V Vapnik and A Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24, 1963.

[97] A. Verikas, A. Gelzinis, and M. Bacauskiene. Mining data with random forests: A survey and results of new tests. *Pattern Recognition*, 44(2):330–349, feb 2011.

[98] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, pages 1096–1103. ACM, 2008.

[99] Huaqing Wang and Peng Chen. A feature extraction method based on information theory for fault diagnosis of reciprocating machinery. *Sensors*, 9(4):2415–2436, apr 2009.

[100] I. Wattar, W. Hafez, and Z. Gao. Model-based diagnosis of chaotic vibration signals. *IECON'99. Conference Proceedings. 25th Annual Conference of the IEEE Industrial Electronics Society (Cat. No.99CH37029)*, 1999.

[101] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

[102] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[103] Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4*, pages 96–104, New York, 1960. IRE.

[104] Ruqiang Yan, Robert X. Gao, and Xuefeng Chen. Wavelets for fault diagnosis of rotary machines: A review with applications. *Signal Processing*, 96:1–15, mar 2014.

[105] Bo-Suk Yang, Xiao Di, and Tian Han. Random forests classifier for machine fault diagnosis. *Journal of Mechanical Science and Technology*, 22(9):1716–1725, sep 2008.

[106] Han Lun Yap and Christopher J. Rozell. Stable takens embedding for linear dynamical systems. In *49th IEEE Conference on Decision and Control (CDC)*. IEEE, dec 2010.

[107] Yang Yu, YuDejie, and Cheng Junsheng. A roller bearing fault diagnosis method based on EMD energy entropy and ANN. *Journal of Sound and Vibration*, 294(1-2):269–277, jun 2006.

[108] Jing ZHOU, Yong QIN, Linlin KOU, Mitchell YUWONO, and Steven SU. Fault detection of rolling bearing based on FFT and classification. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 9(5):JAMDSM0056–JAMDSM0056, 2015.